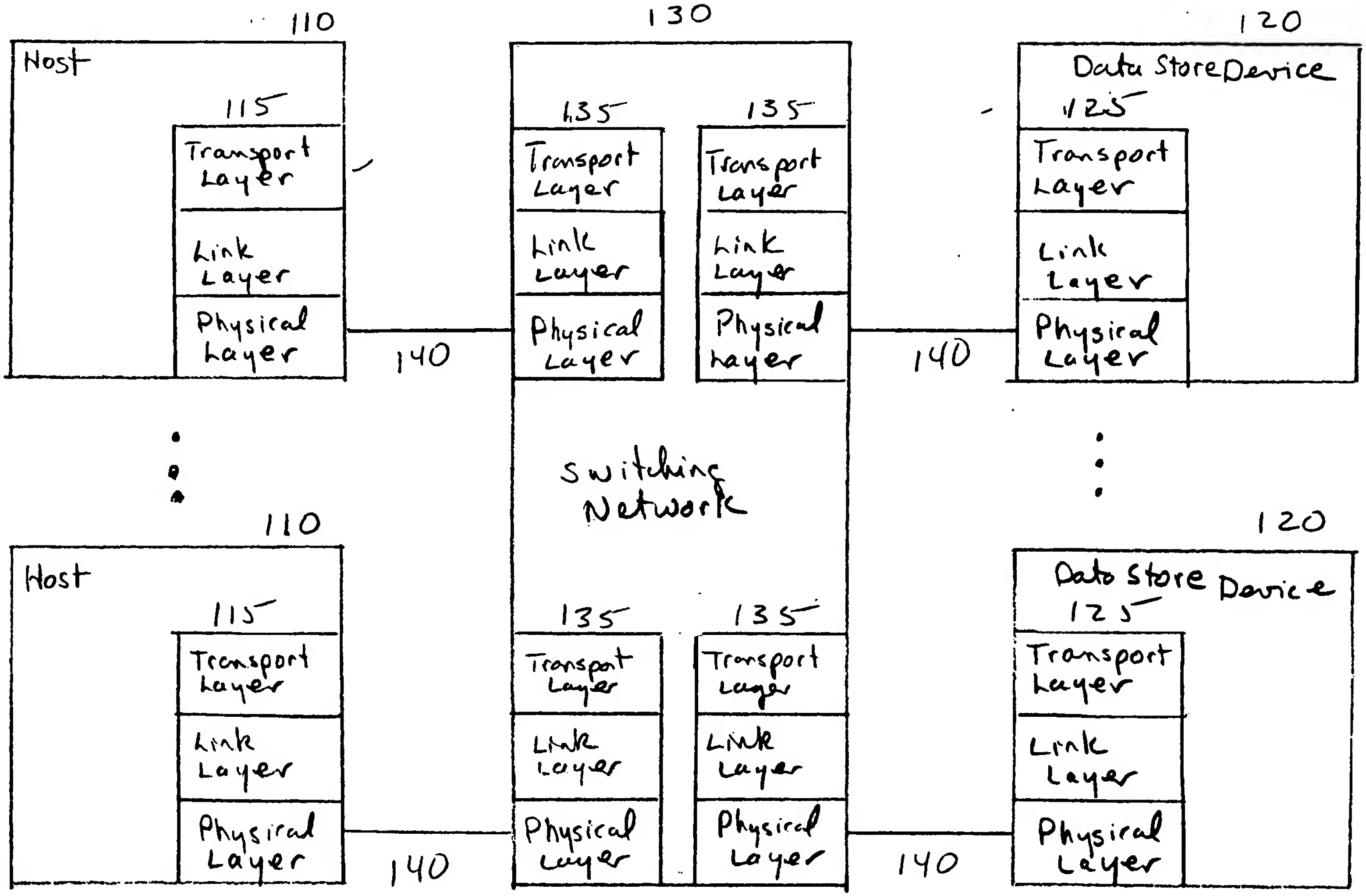

110
Host
115
Transport Layer
Link Layer
Physical Layer
110
Host
115
Transport Layer
Link Layer
Physical Layer
140
140
130
135
Transport Layer
Link Layer
Physical Layer
135
Transport Layer
Link Layer
Physical Layer
Switching Network
135
Transport Layer
Link Layer
Physical Layer
135
Transport Layer
Link Layer
Physical Layer
140
140
120
Data Store Device
125
Transport Layer
Link Layer
Physical Layer
120
Data Store Device
125
Transport Layer
Link Layer
Physical Layer

Fig 1

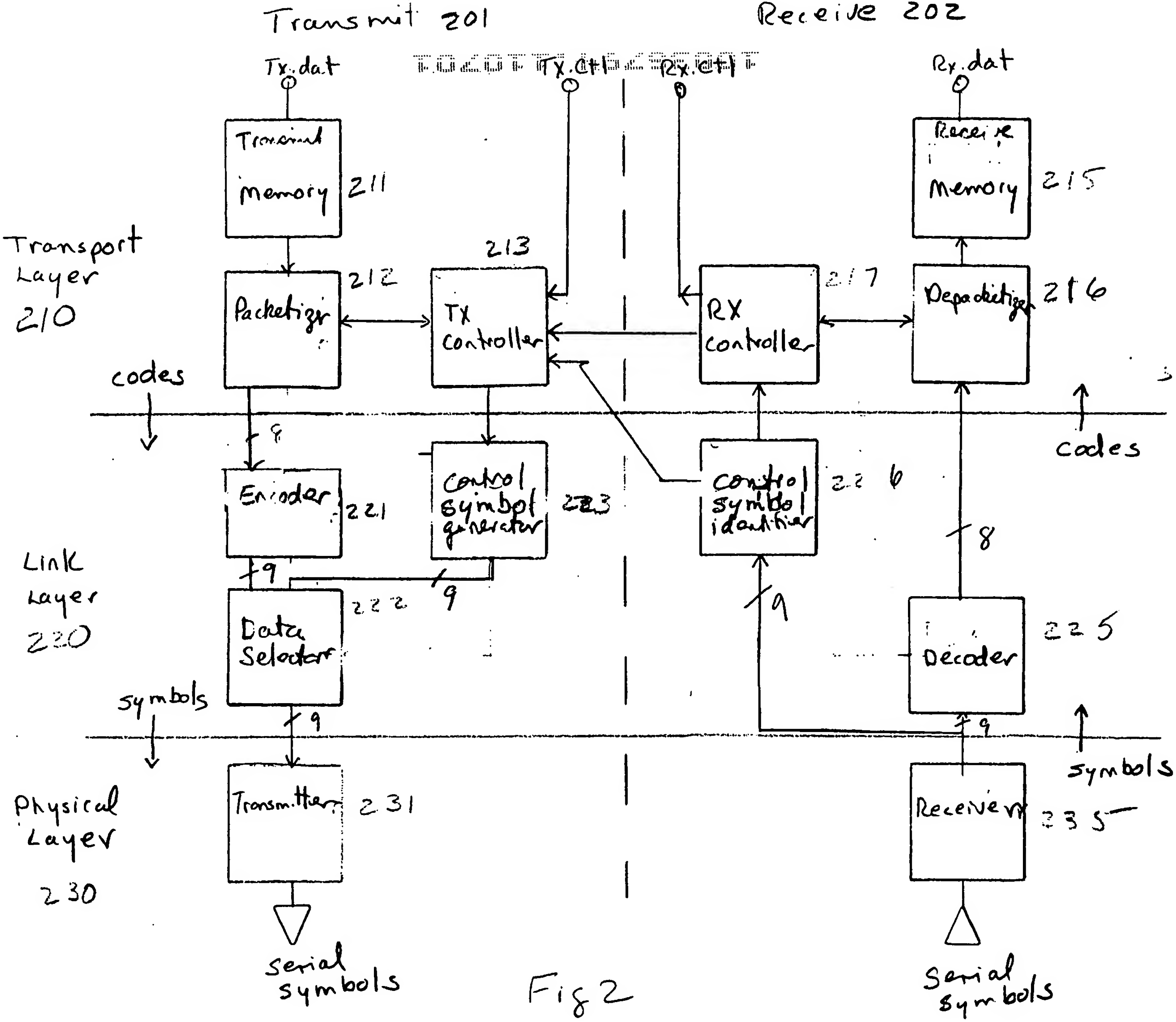

Transmit 201
Receive 202
Tx.dat
Tx.ctl
Rx.ctl
Rx.dat
Transmit memory
211
Receive memory
215
Transport Layer 210
Packetizer
212
TX controller
213
RX controller
217
Depacketizer
216
codes
8
codes
Encoder
221
control symbol generator
223
control symbol identifier
226
8
Link Layer 220
9
9
Data Selector
222
9
Decoder
225
symbols
9
9
symbols
Transmitter
231
Receiver
235
Physical Layer 230
serial symbols
Serial symbols

Fig 2

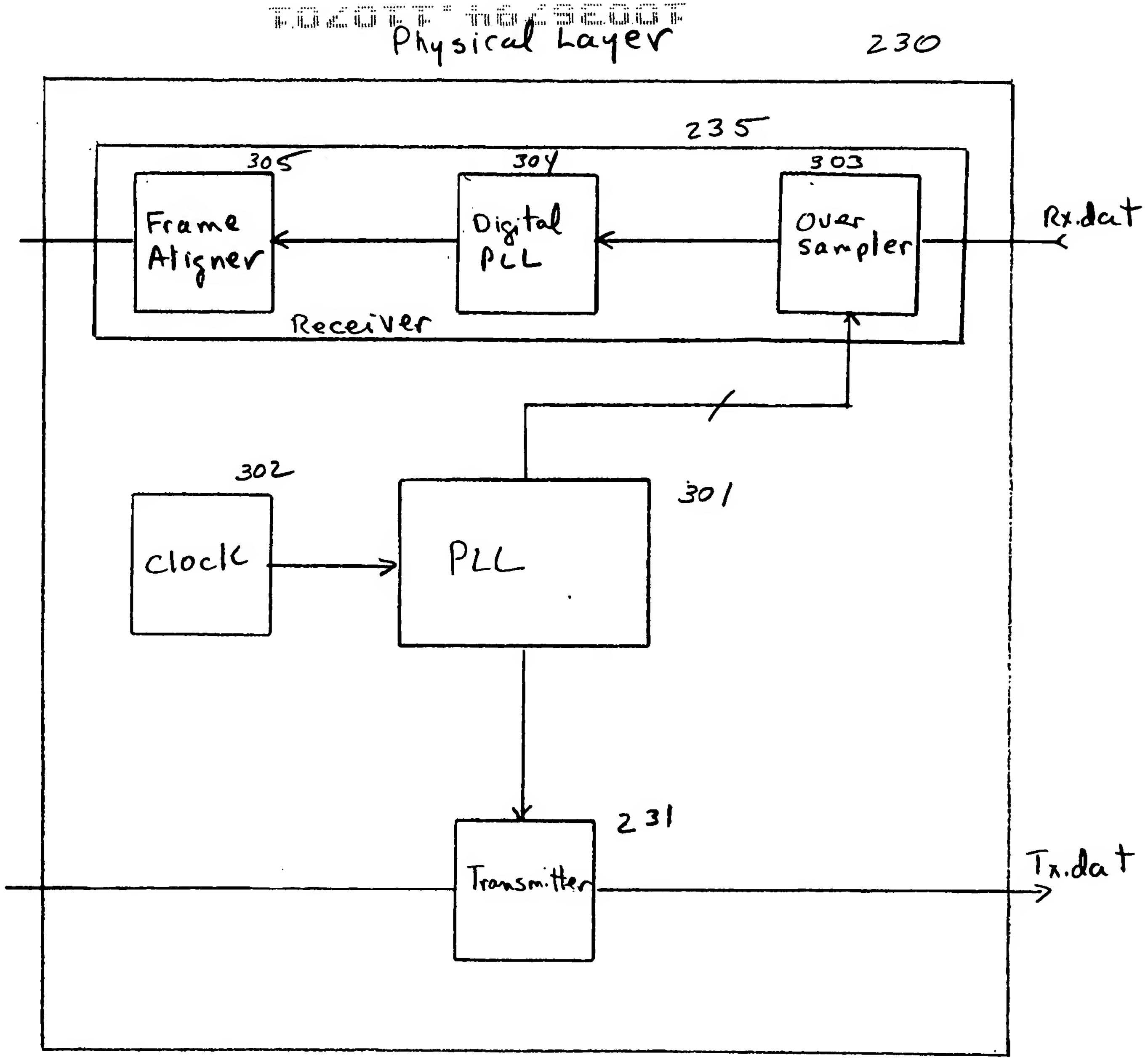

Physical Layer
230
235
305
304
303
Frame Aligner
Digital PLL
Over Sampler
Rx.dat
Receiver
302
Clock
PLL
301
231
Transmitter
Tx.dat

Fig 3

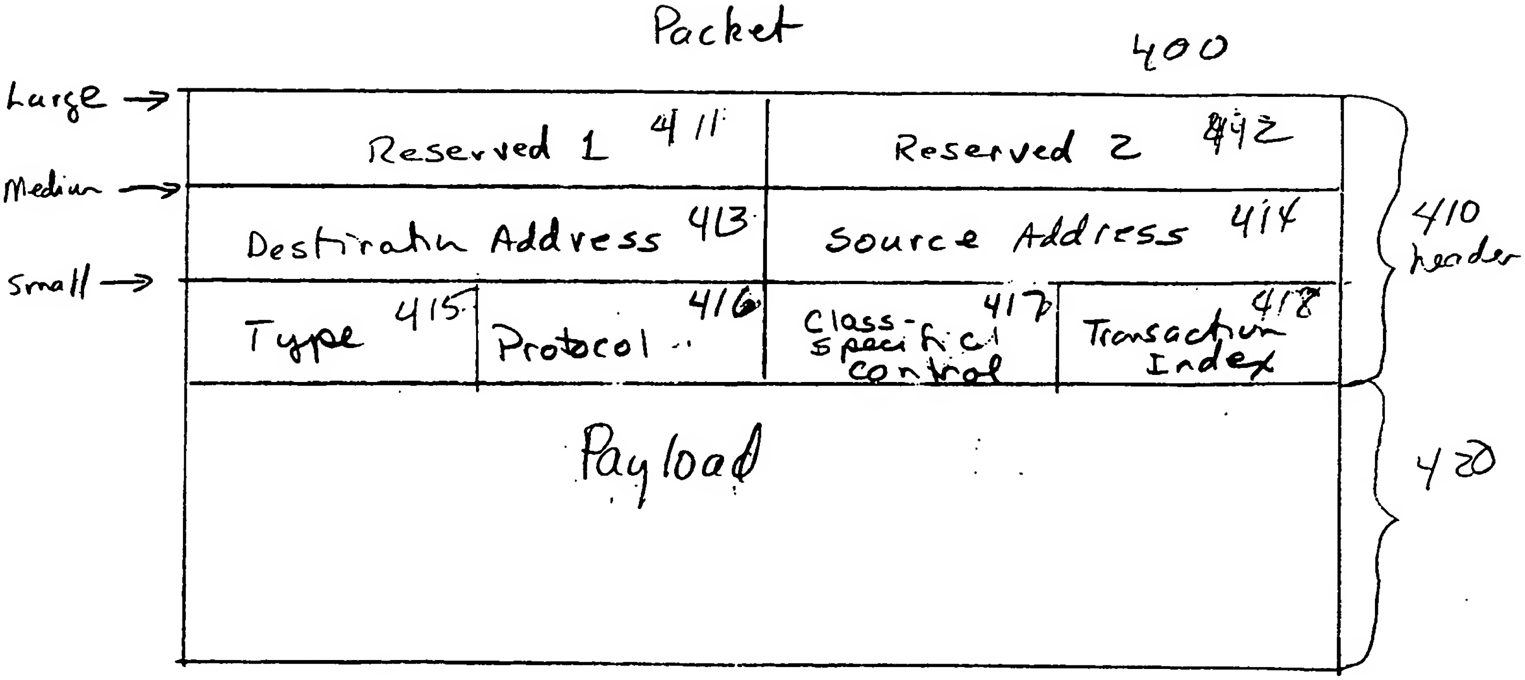

Packet
400
Large →
Medium →
Small →
Reserved 1 411
Reserved 2 412
Destination Address 413
Source Address 414
Type 415
Protocol 416
Class-specific control 417
Transaction Index 418
410 header
Payload
420

Fig 4

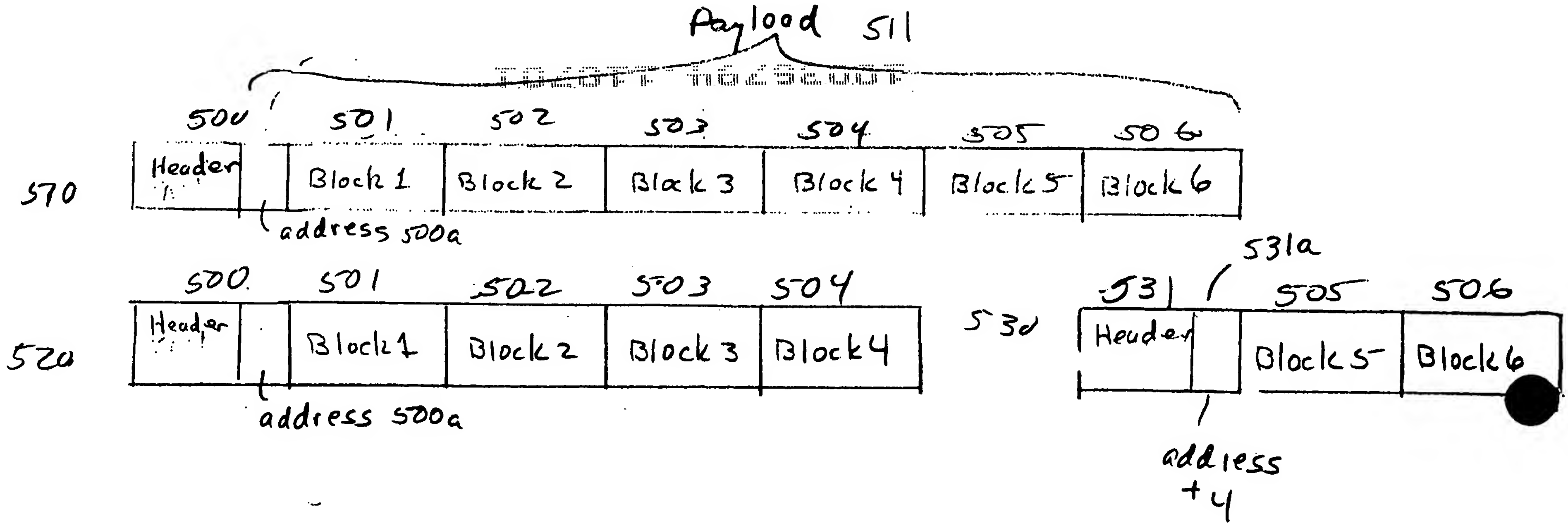

Payload 511
510
500
501
502
503
504
505
506
Header
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
address 500a
520
500
501
502
503
504
Header
Block1
Block2
Block3
Block4
address 500a
530
531
531a
505
506
Header
Block5
Block6
address +4

Fig 5

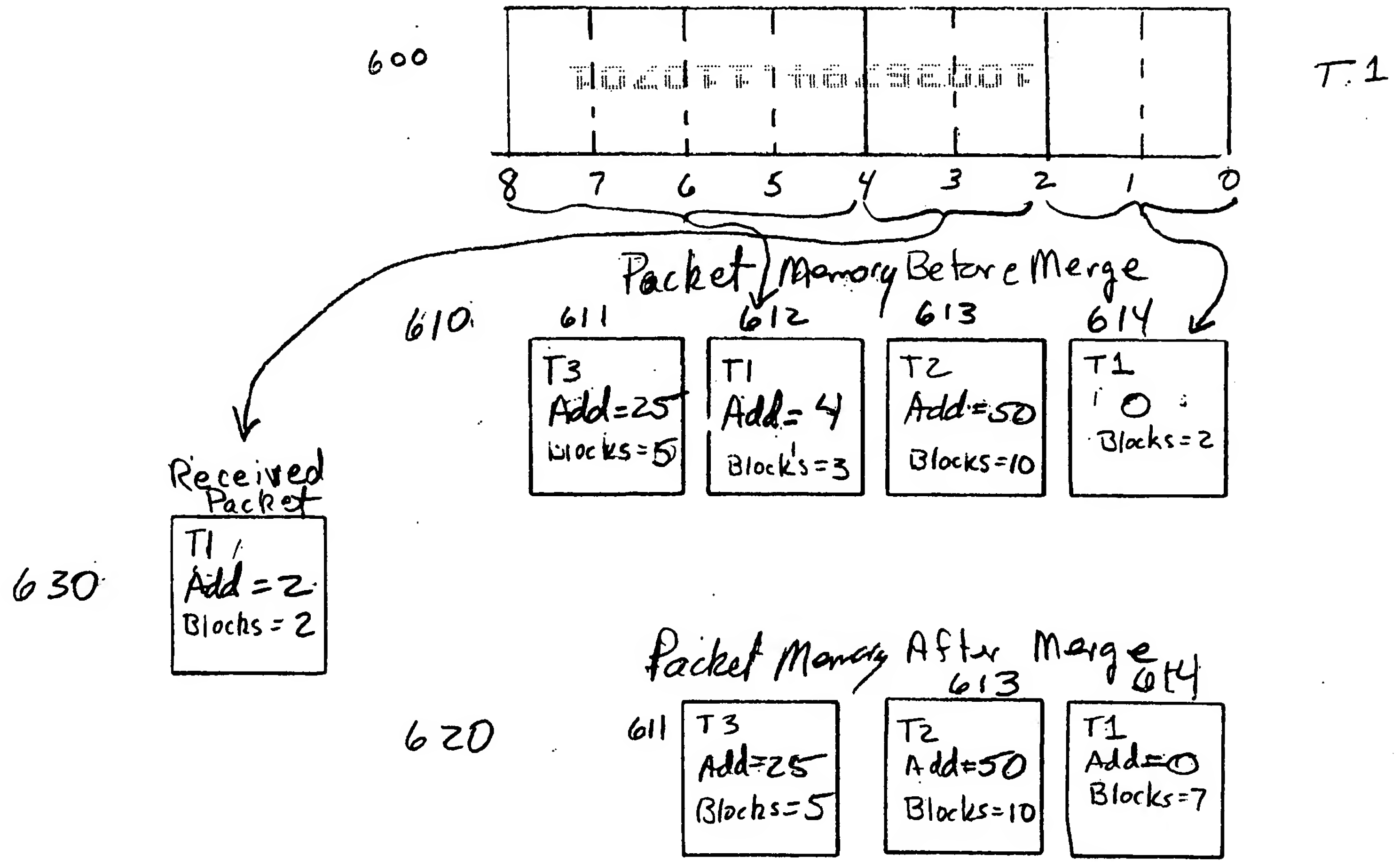

600
T.1
8
7
6
5
4
3
2
1
0
Packet Memory Before Merge
610
611
T3
Add=25
Blocks=5
612
T1
Add=4
Blocks=3
613
T2
Add=50
Blocks=10
614
T1
O
Blocks=2
Received Packet
630
T1
Add=2
Blocks=2
Packet Memory After Merge
620
611
T3
Add=25
Blocks=5
613
T2
Add=50
Blocks=10
614
T1
Add=0
Blocks=7

Fig 6

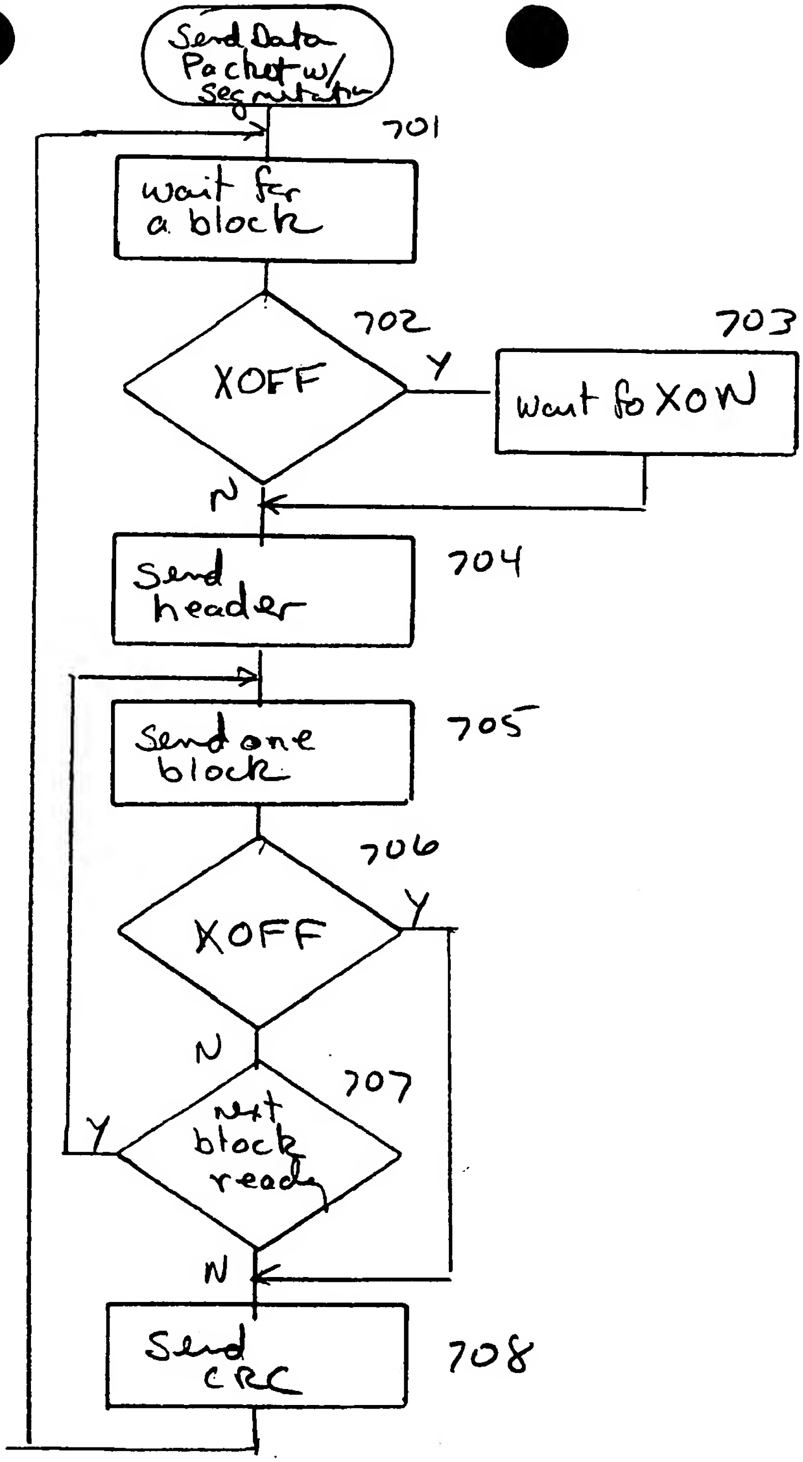
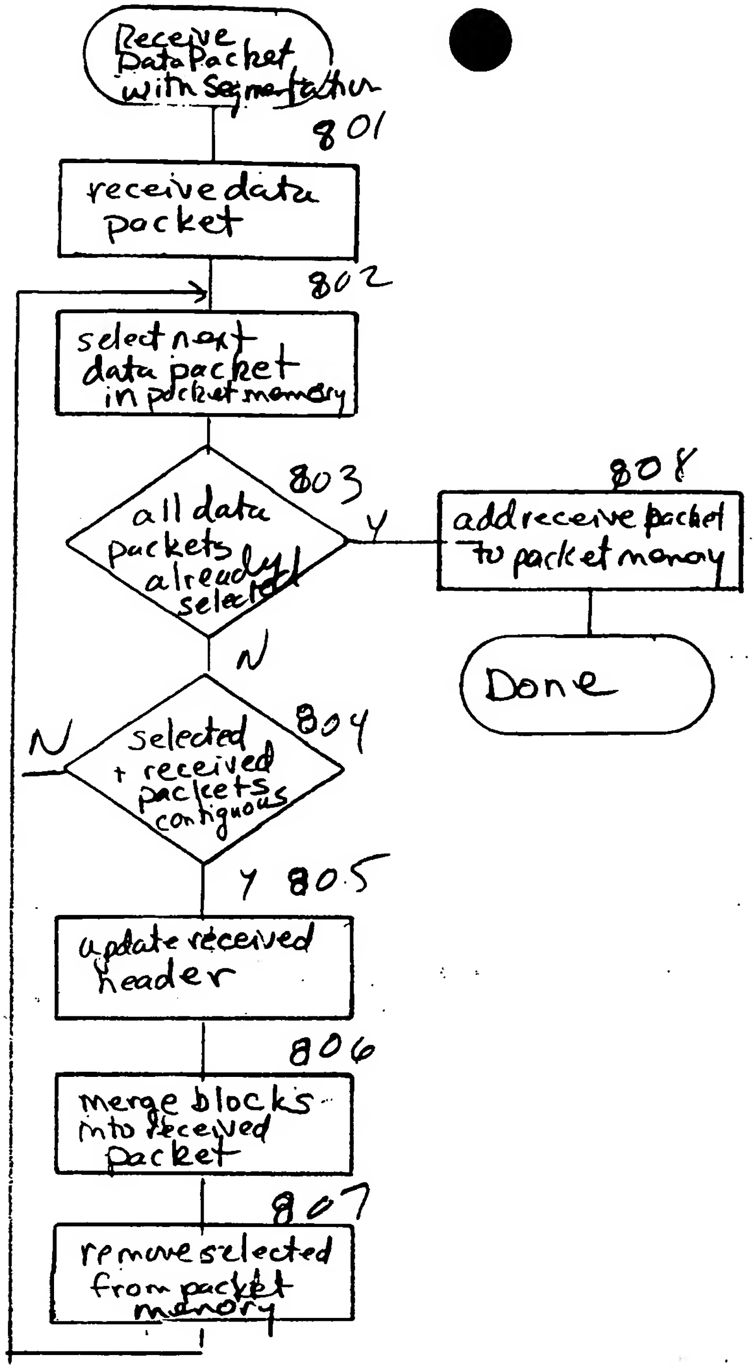
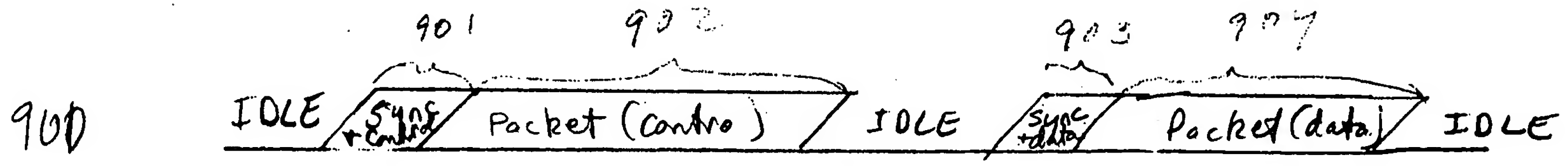

Send Data Packet w/ segmentation
701
wait for a block
702
XOFF
Y
703
wait for XON
N
704
Send header
705
Send one block
706
XOFF
Y
N
707
next block ready
Y
N
708
Send CRC

Fig 7

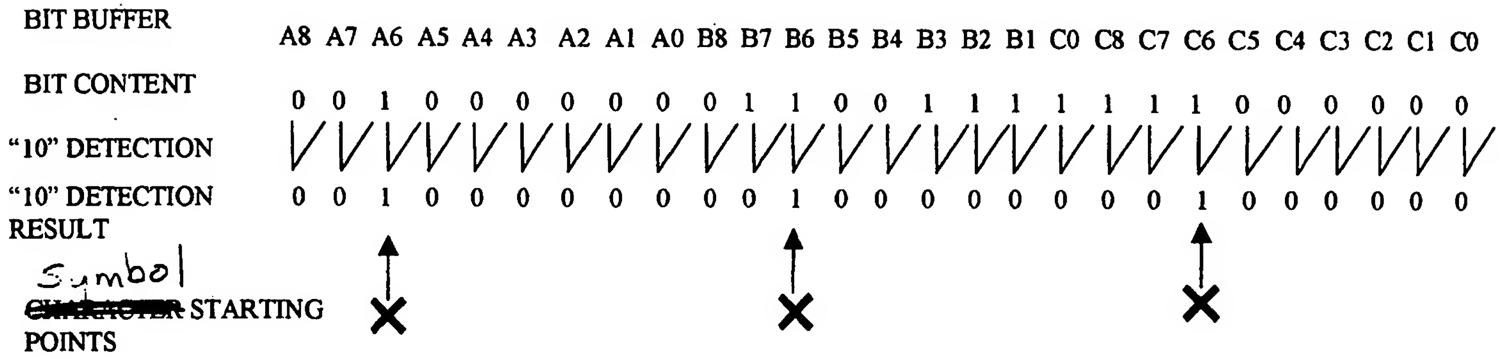

Receive DataPacket with Segmentation
801
receive data packet
802
select next data packet in packet memory
803
all data packets already selected
Y
808
add receive packet to packet memory
Done
N
N
804
selected + received packets contiguous
Y 805
update received header
806
merge blocks into received packet
807
remove selected from packet memory

Fi 8



Sync + packet type

Fig 9A



BIT BUFFER	A8	A7	A6	A5	A4	A3	A2	A1	A0	B8	B7	B6	B5	B4	B3	B2	B1	C0	C8	C7	C6	C5	C4	C3	C2	C1	C0
BIT CONTENT	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0
“10” DETECTION																											
“10” DETECTION RESULT	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Symbol ~~CHARACTER~~ STARTING POINTS			✕									✕									✕						

FIG.10

Fig 9B

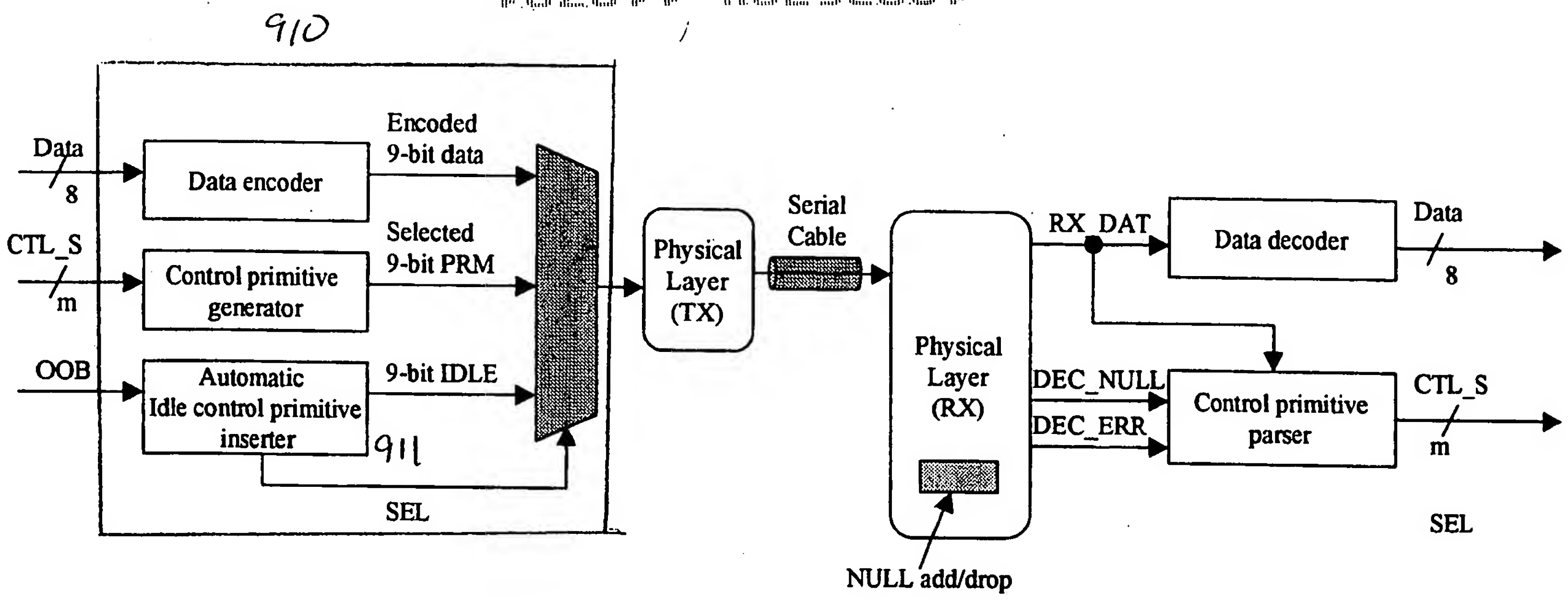

910
Data
8
CTL_S
m
OOB
Data encoder
Control primitive generator
Automatic Idle control primitive inserter
Encoded 9-bit data
Selected 9-bit PRM
9-bit IDLE
911
SEL
Physical Layer (TX)
Serial Cable
Physical Layer (RX)
NULL add/drop
RX_DAT
DEC_NULL
DEC_ERR
Data decoder
Control primitive parser
Data
8
CTL_S
m
SEL

Fig. 9C

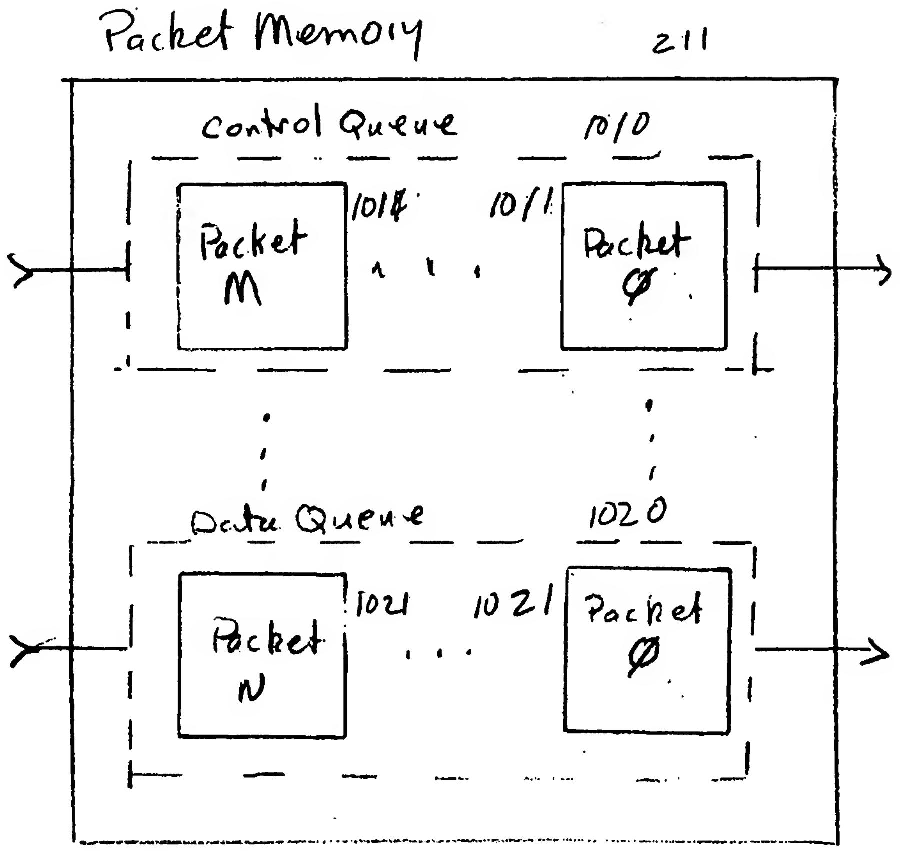

Packet Memory
211
Control Queue
1010
1014
1011
Packet M
Packet Ø
Data Queue
1020
1021
1021
Packet N
Packet Ø

Fig 10

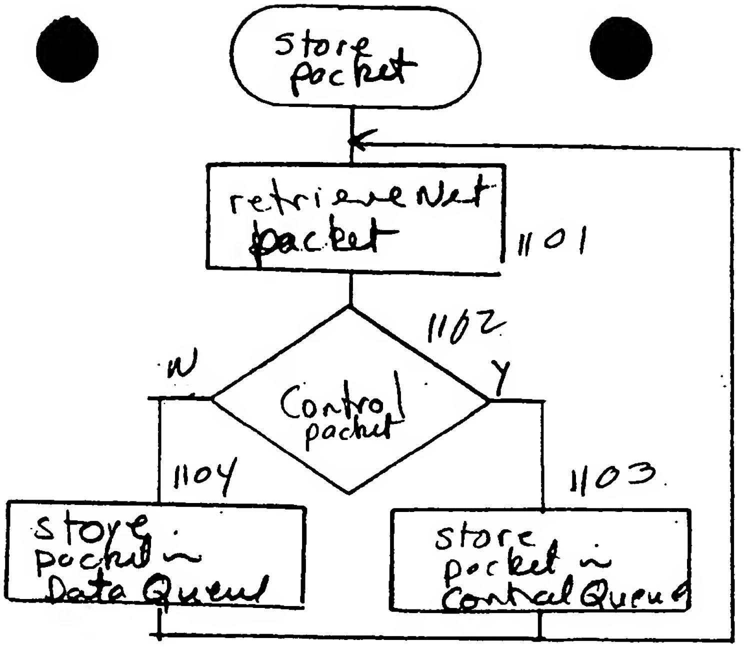

store packet
retrieve Net packet
1101
1102
Control packet
N
Y
1104
1103
store packet in Data Queue
store packet in Control Queue

Fig 11

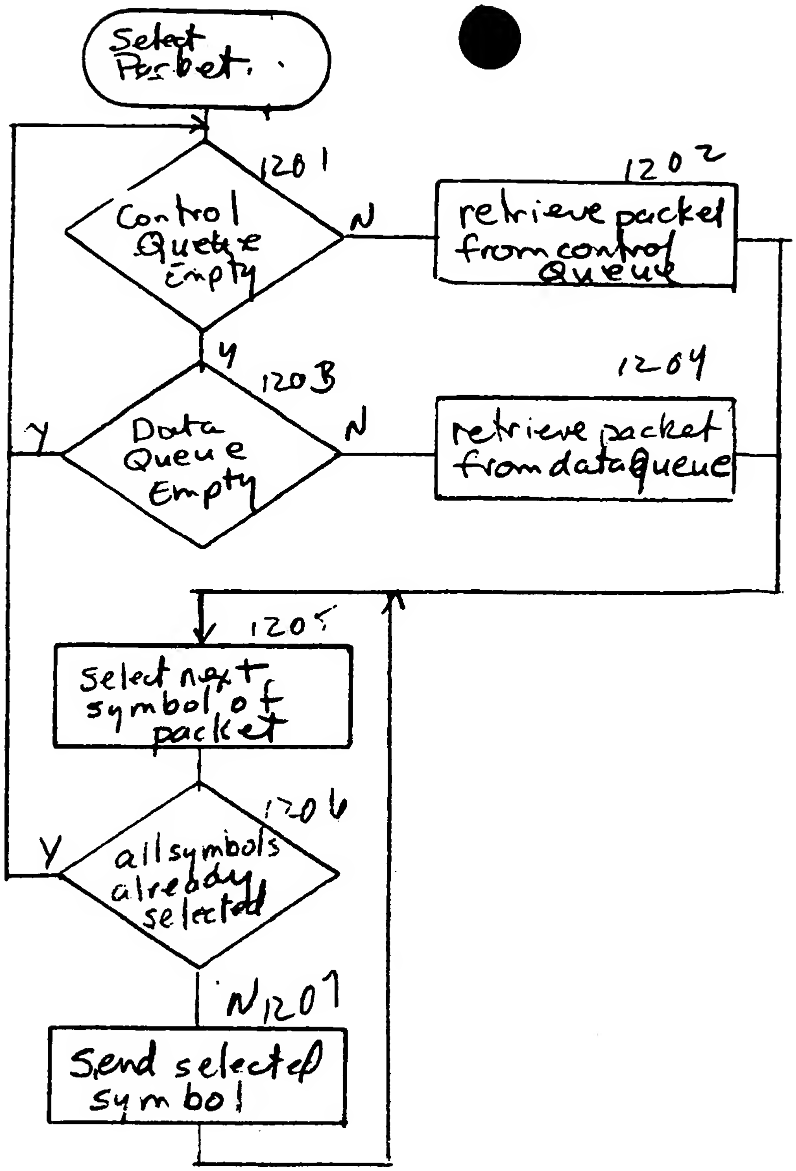

Select Packet.
Control Queue empty
1201
N
retrieve packet from control Queue
1202
Y
Data Queue Empty
1203
N
retrieve packet from dataqueue
1204
Y
select next symbol of packet
1205
all symbols already selected
1206
Y
N
1207
Send selected symbol

Fig 12

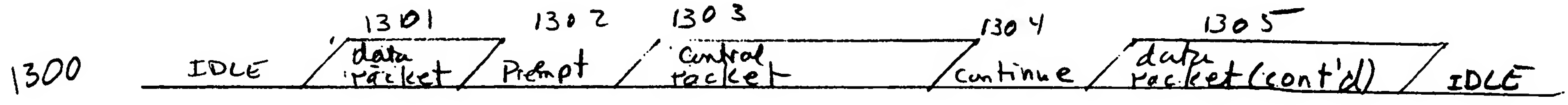

1300
IDLE
1301
data packet
1302
Prempt
1303
control packet
1304
continue
1305
data packet (cont'd)
IDLE

Fig 13

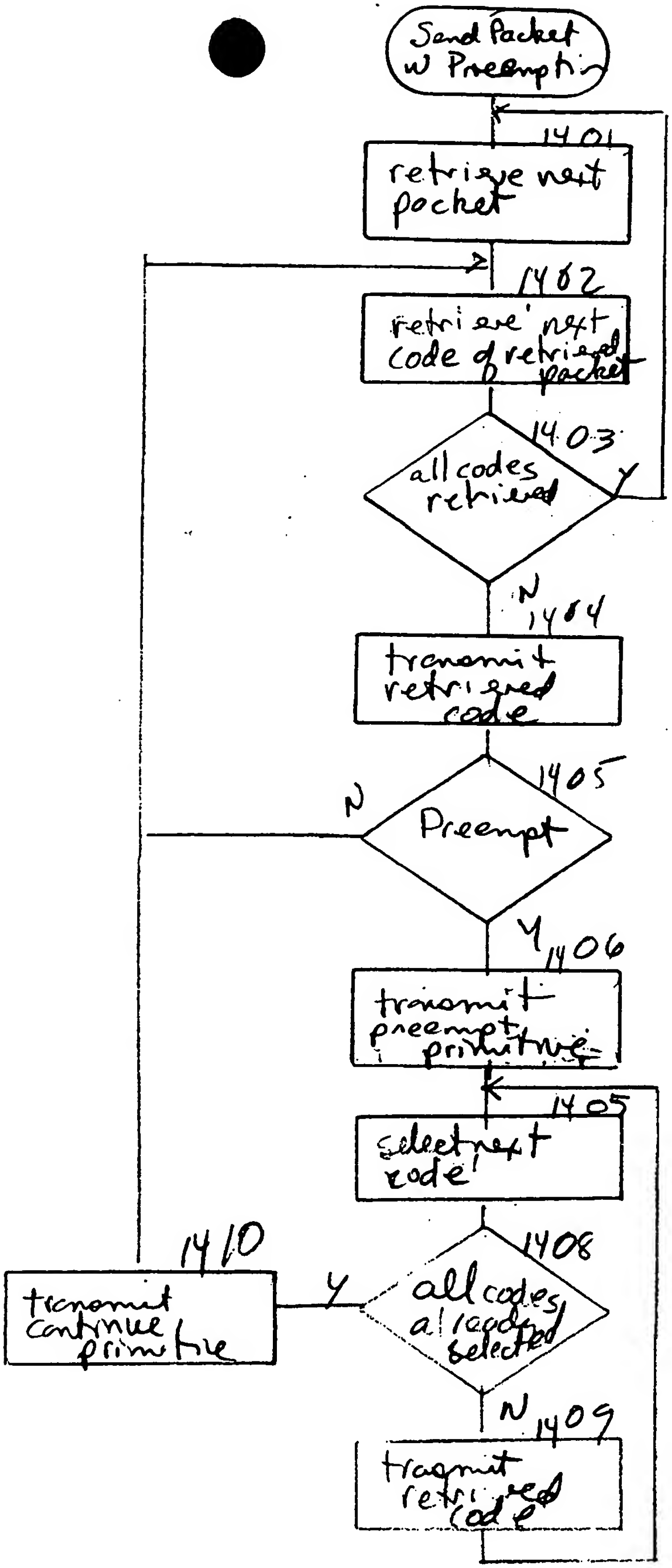

Send Packet w Preemption
1401
retrieve next pocket
1402
retrieve next code of retrieved packet
1403
all codes retrieved
Y
N
1404
transmit retrieved code
1405
Preempt
N
Y
1406
transmit preempt primitive
1405
select next code
1408
all codes already selected
Y
N
1410
transmit continue primitive
1409
transmit retrieved code

Fig 14

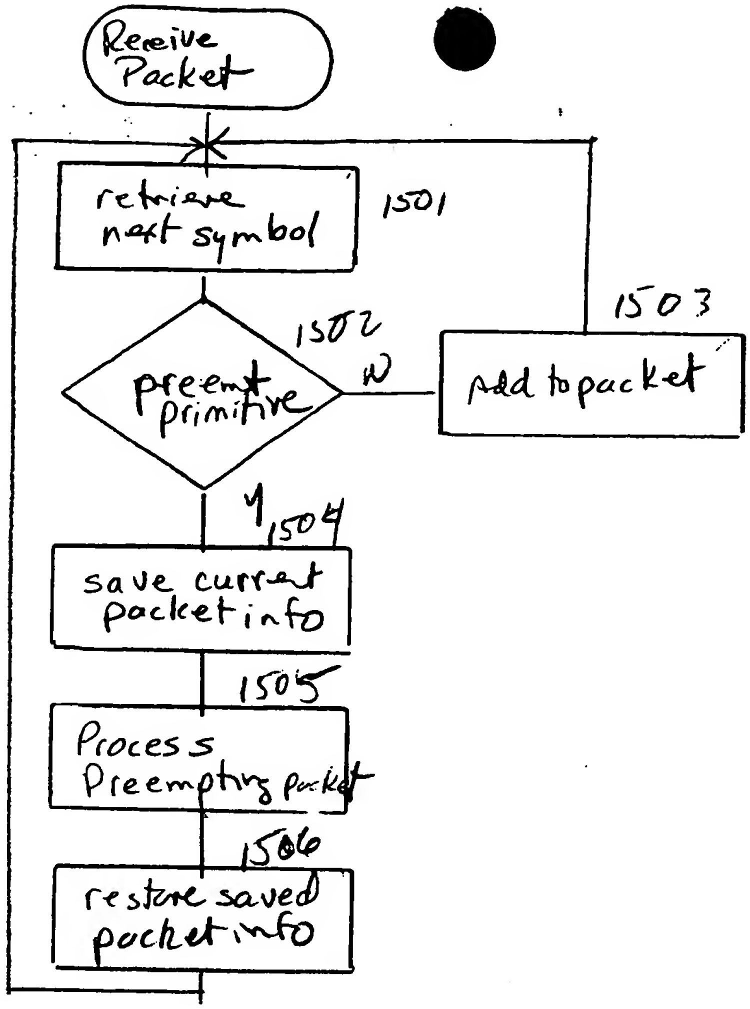

Receive Packet
retrieve next symbol
1501
preemt primitive
1502
N
Add to packet
1503
Y
1504
save current packet info
1505
Process Preempting packet
1506
restore saved packet info

Fig 15

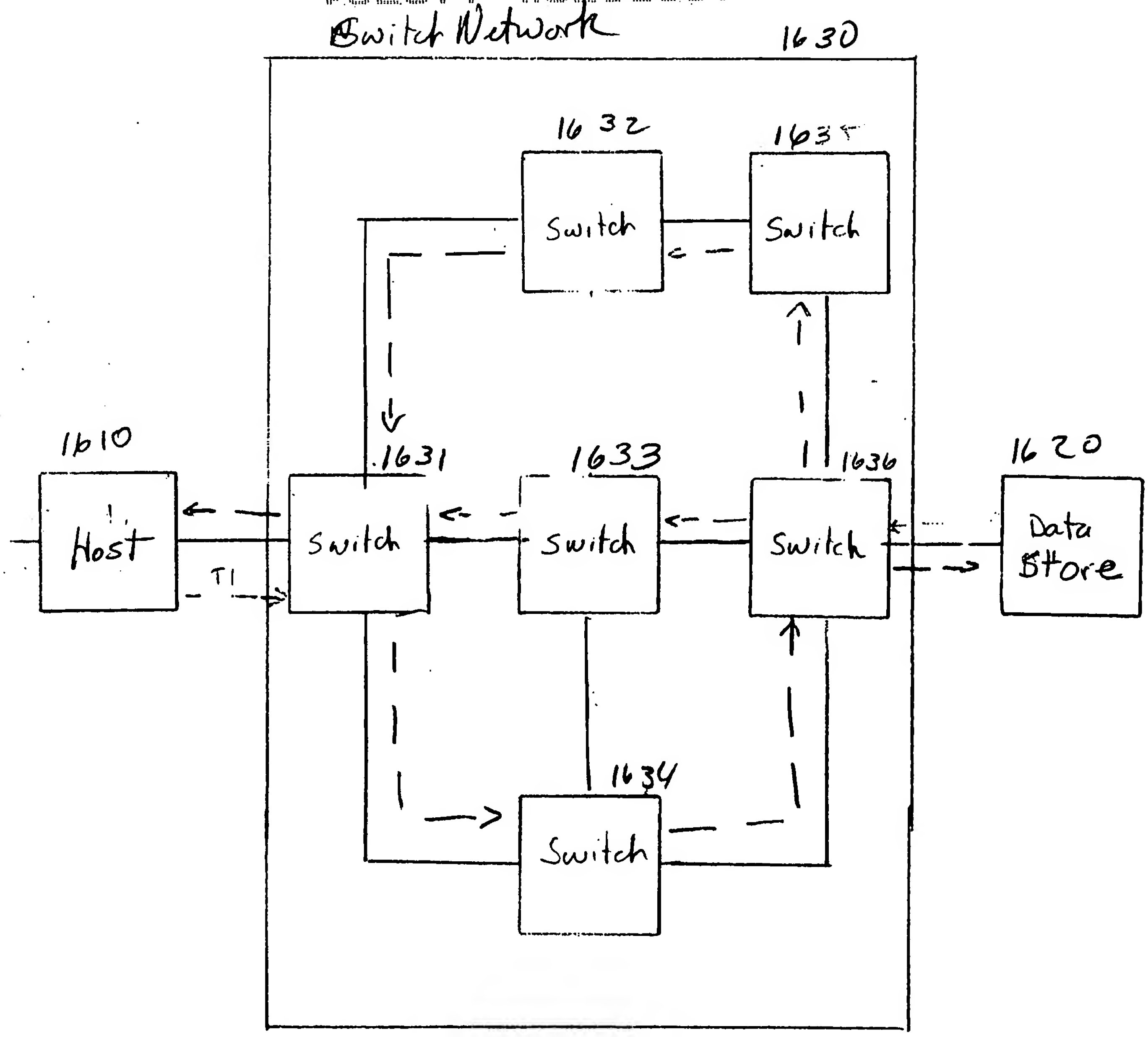
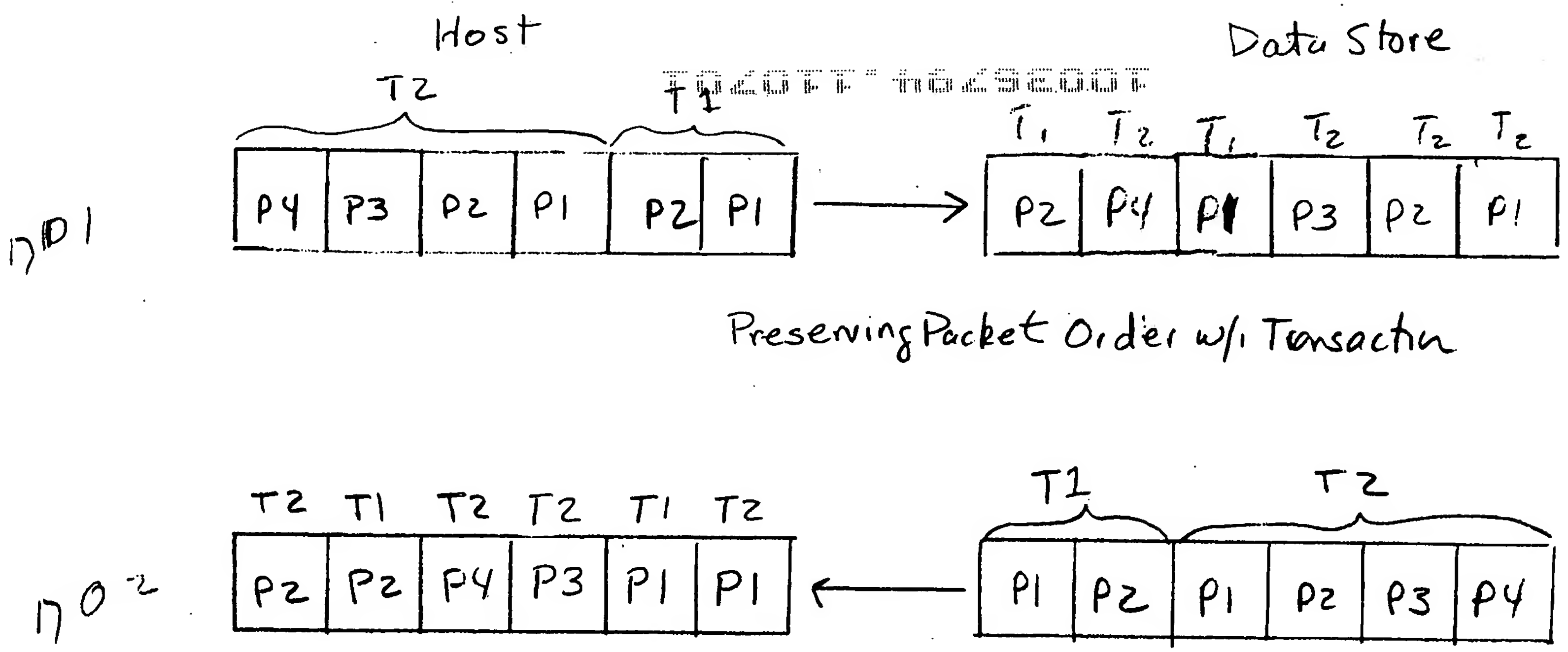

Switch Network
1630
1632
Switch
1637
Switch
1610
Host
T1
1631
Switch
1633
Switch
1636
Switch
1620
Data
Store
1634
Switch

Fig 16



Preserving Packet Order w/i Transactin

No Packet or TransactinOrdering

Fig 17

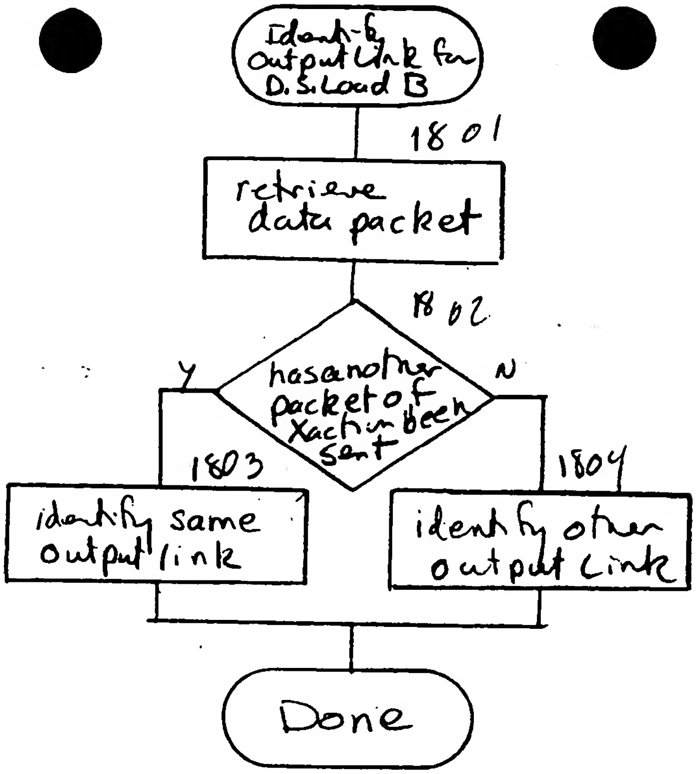
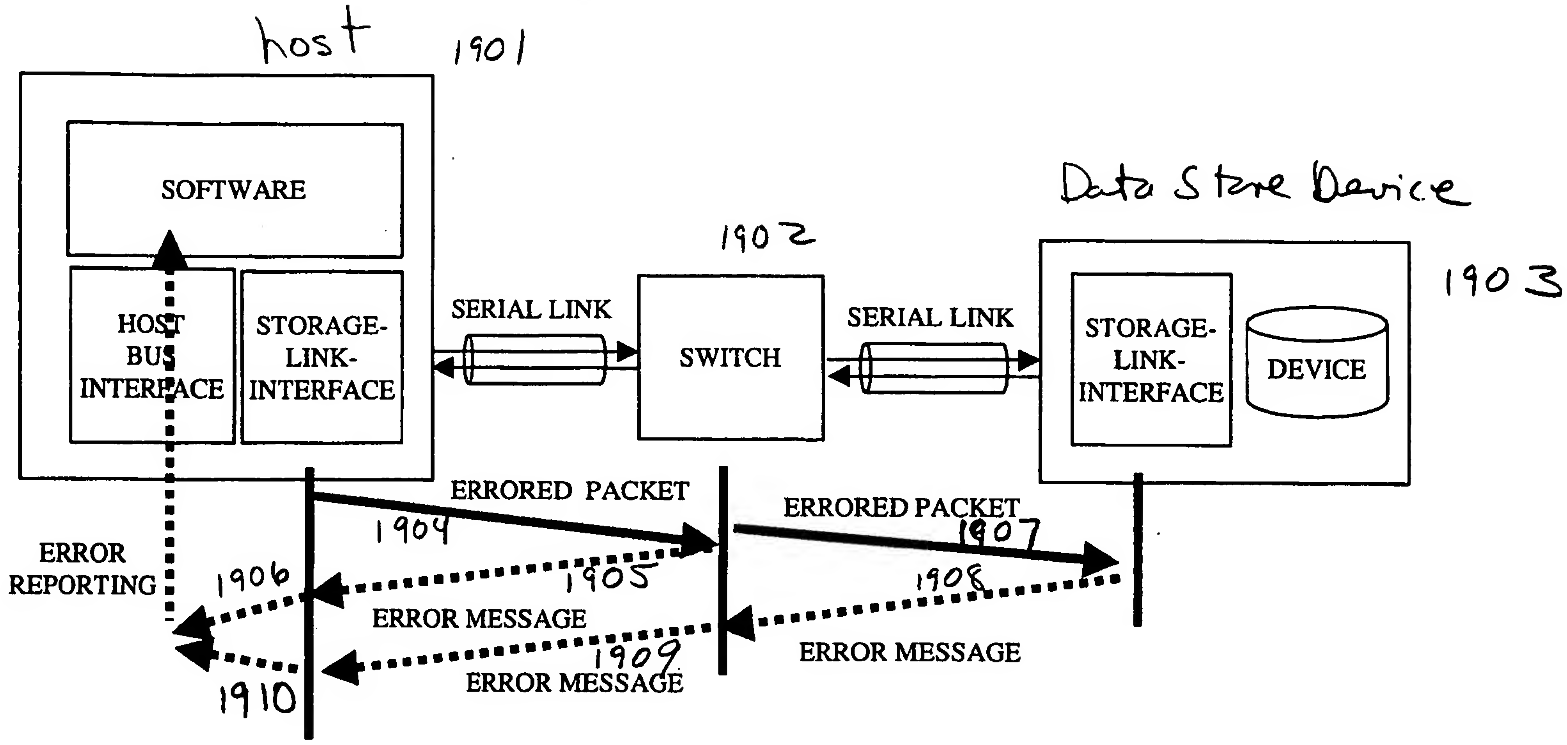

Identify Output Link for D.S. Load B
1801
retrieve data packet
1802
has another packet of Xaction been sent
Y
N
1803
identify same output link
1804
identify other output Link
Done

Fig 18



~~FIG.19~~

Fig 19A

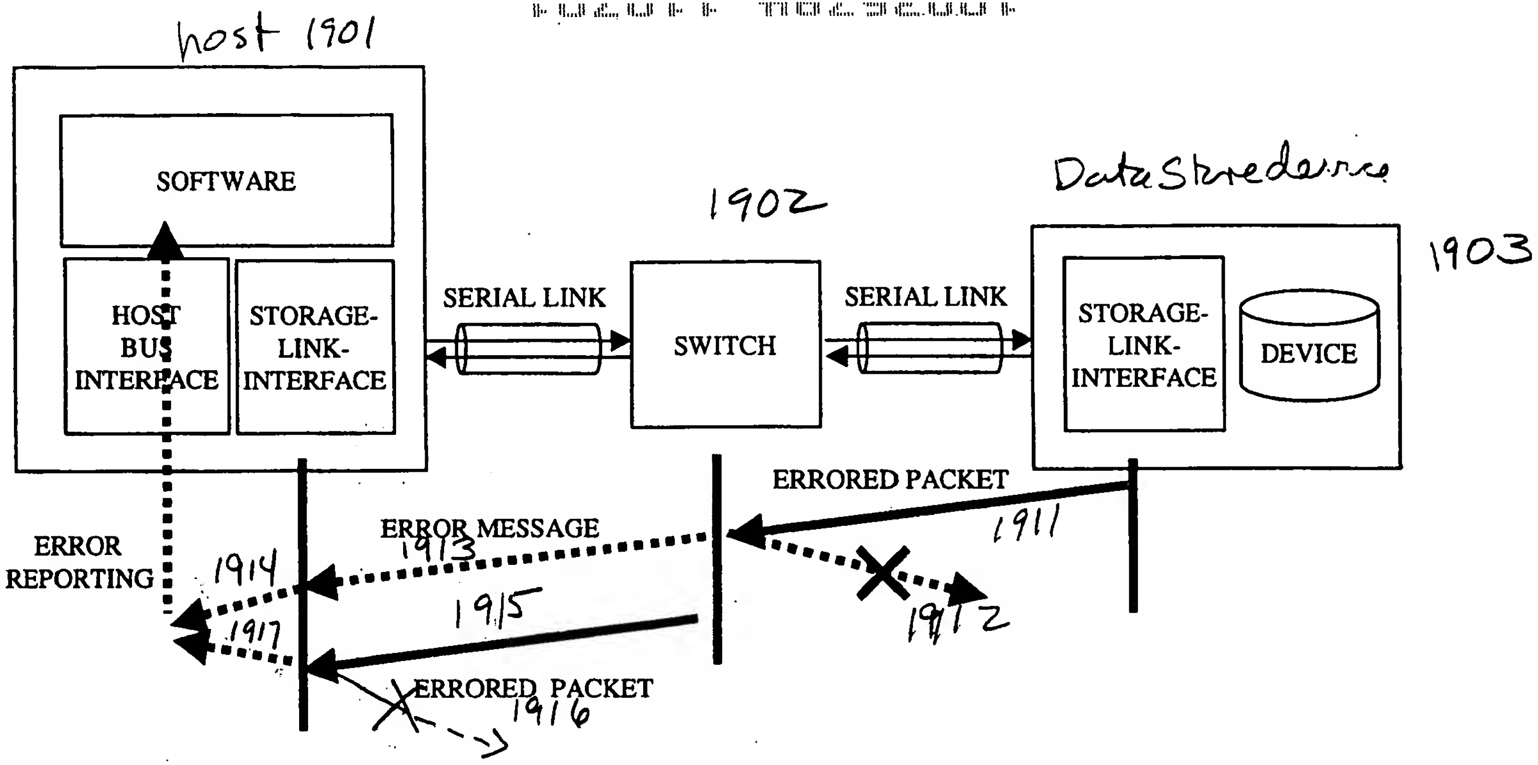
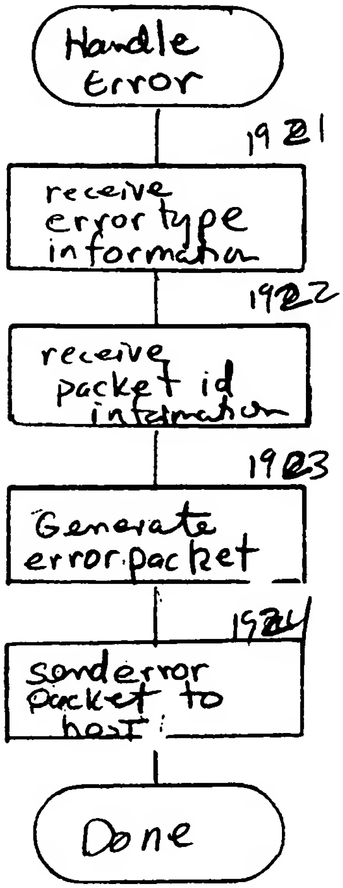


Fig 19B

~~FIG 20~~


Handle Error
1921
receive error type information
1922
receive packet id information
1923
Generate error packet
1924
send error packet to host
Done

19C

8b Code	9 bit symbol
0000 0000	101010101
0000 0001	101010100
0000 0010	101010111
⋮	
0101 0101	001010101
⋮	
0111 0110	001110110
0111 0111	100100010
⋮	
1111 1111	110101010

Fig 20

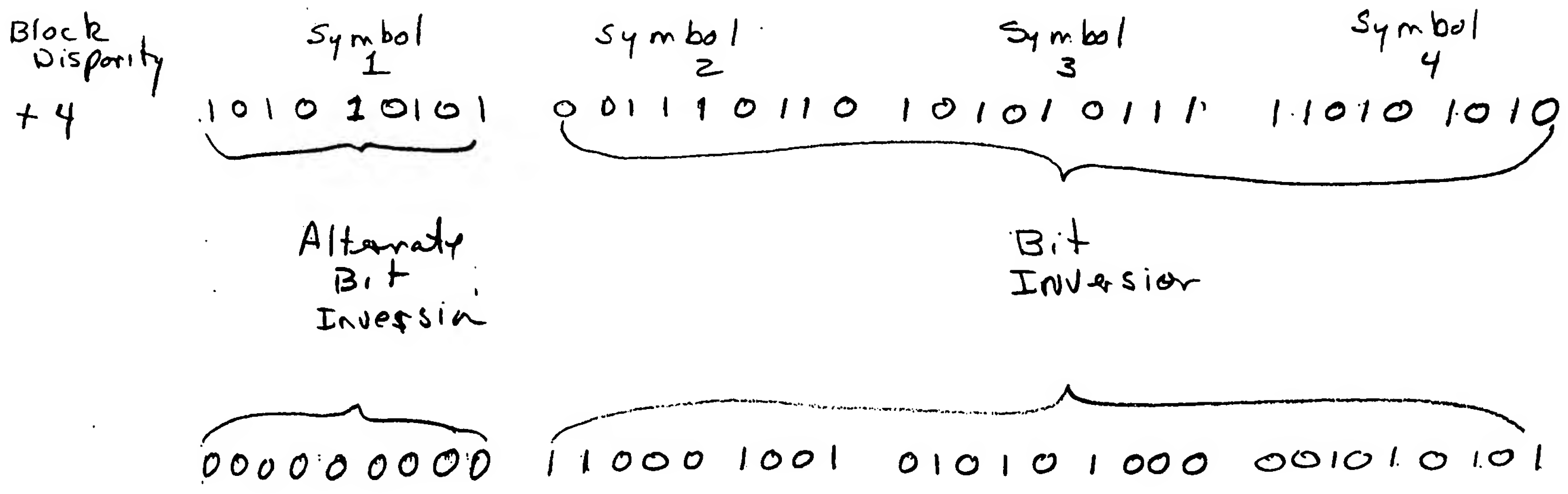

Block Disparity
+4
Symbol 1
1010 10101
Symbol 2
0 01110110
Symbol 3
10101 0111
Symbol 4
11010 1010
Alternate Bit Inversion
Bit Inversion
000000000
11000 1001
01010 1000
00101 0101

Fig 21A

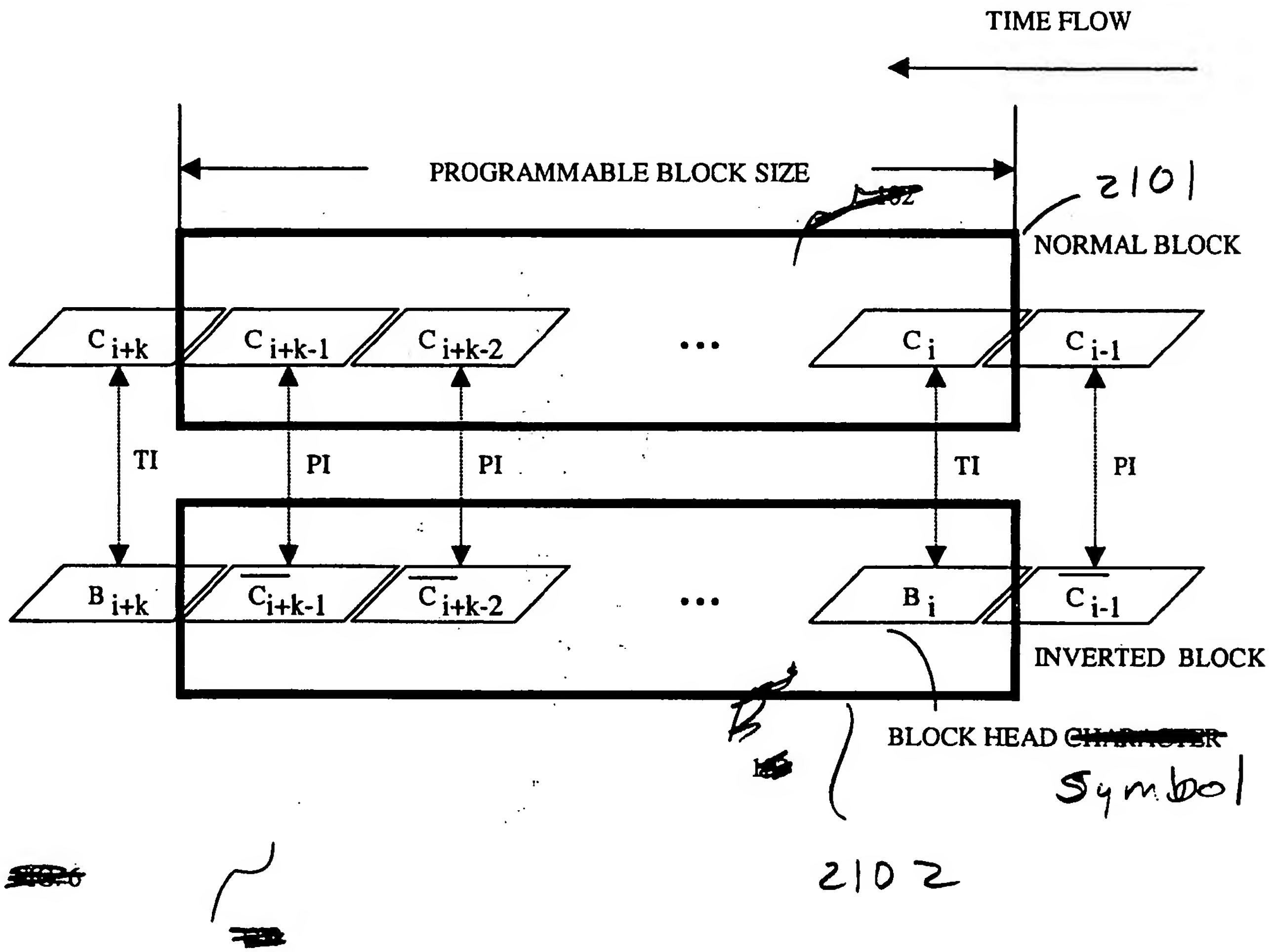
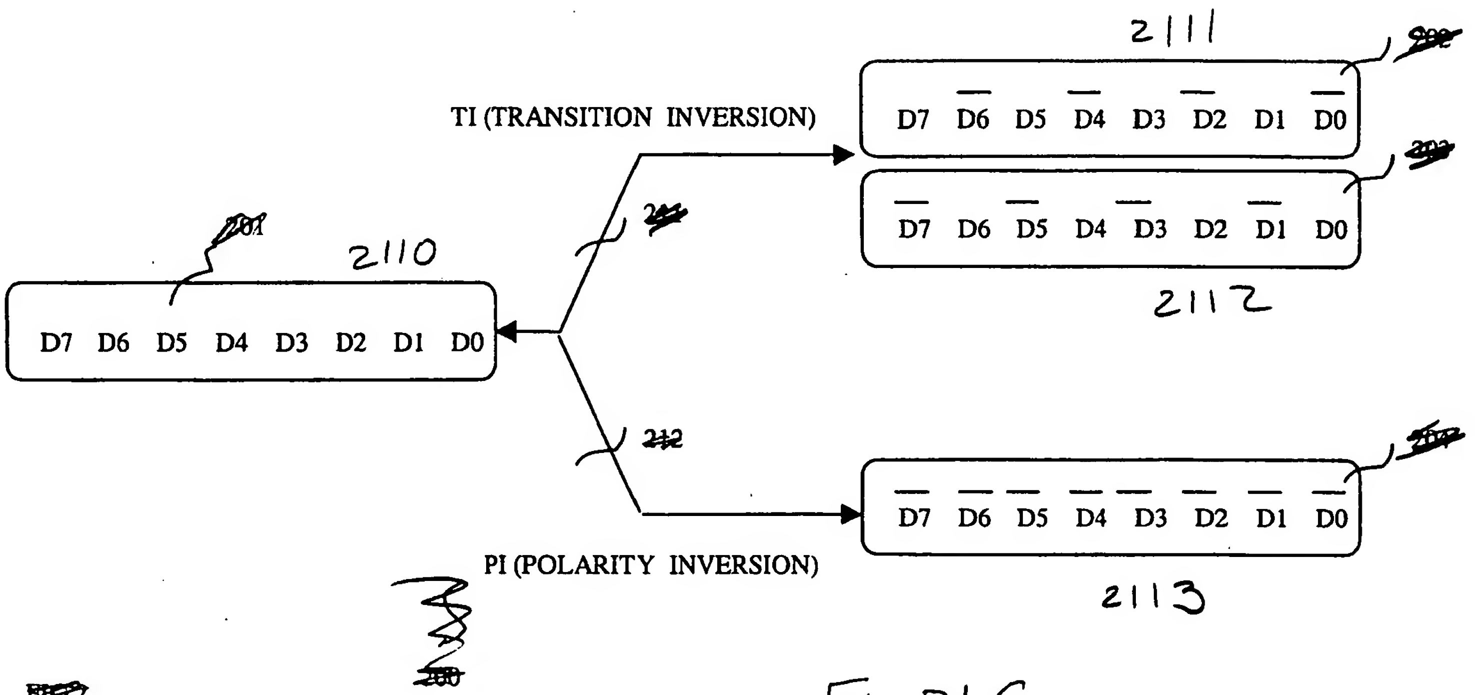

TIME FLOW
PROGRAMMABLE BLOCK SIZE
2101
NORMAL BLOCK
C_{i+k}
C_{i+k-1}
C_{i+k-2}
...
C_i
C_{i-1}
TI
PI
PI
TI
PI
B_{i+k}
$\overline{C}_{i+k-1}$
$\overline{C}_{i+k-2}$
...
B_i
$\overline{C}_{i-1}$
INVERTED BLOCK
BLOCK HEAD ~~CHARACTER~~ Symbol
2102

Fig 21B


2110
~~201~~
D7 D6 D5 D4 D3 D2 D1 D0
TI (TRANSITION INVERSION)
~~211~~
2111
~~202~~
D7 D6 D5 D4 D3 D2 D1 D0
~~203~~
D7 D6 D5 D4 D3 D2 D1 D0
2112
~~212~~
PI (POLARITY INVERSION)
~~204~~
D7 D6 D5 D4 D3 D2 D1 D0
2113
~~200~~

~~FIG.2~~

Fig 21C

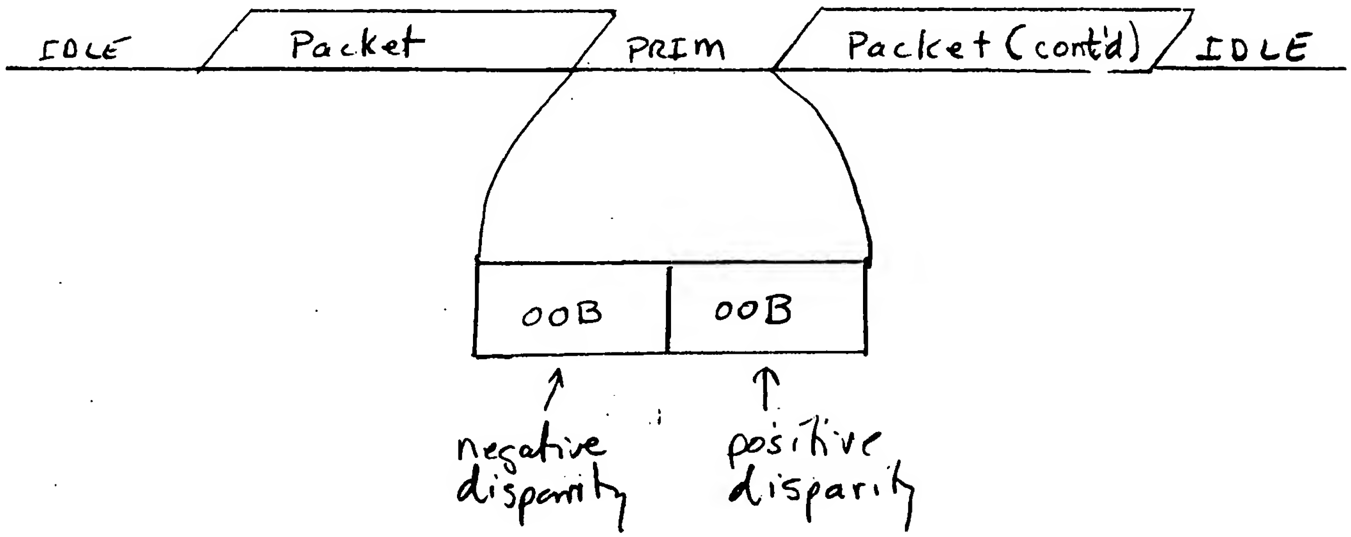

IDLE
Packet
PRIM
Packet (cont'd)
IDLE
OOB
OOB
negative disparity
positive disparity

Fig 22

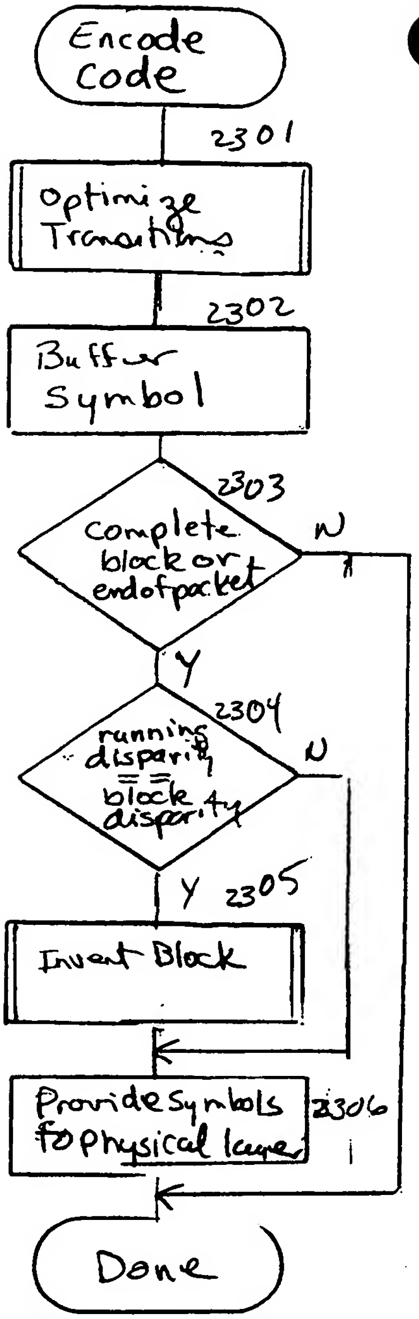

Encode code
2301
Optimize Transitions
2302
Buffer Symbol
2303
complete block or end of packet
N
Y
2304
running disparity == block disparity
N
Y
2305
Invert Block
Provide symbols to physical layer
2306
Done

Fig 23

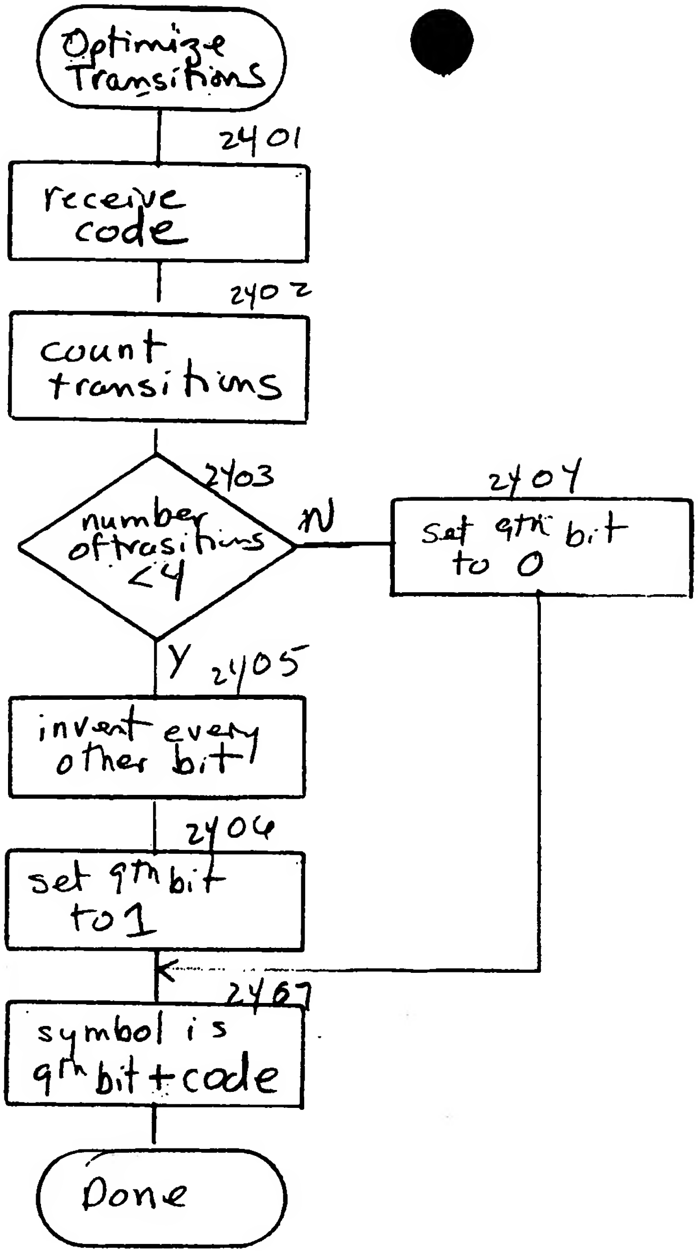

Optimize Transitions
2401
receive code
2402
count transitions
2403
number of trasitions <4
N
2404
set 9th bit to 0
Y
2405
invert every other bit
2406
set 9th bit to 1
2407
symbol is 9th bit + code
Done

Fig 24

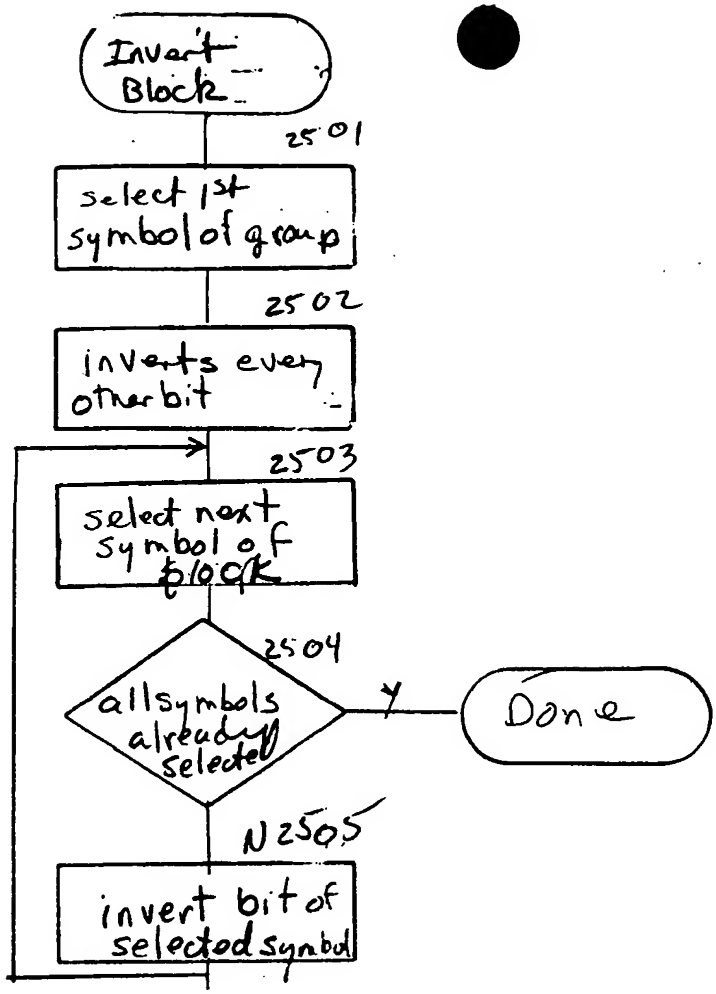

Invert Block
2501
select 1st symbol of group
2502
inverts every other bit
2503
select next symbol of block
2504
all symbols already selected
Y
Done
N 2505
invert bit of selected symbol

Fig 25

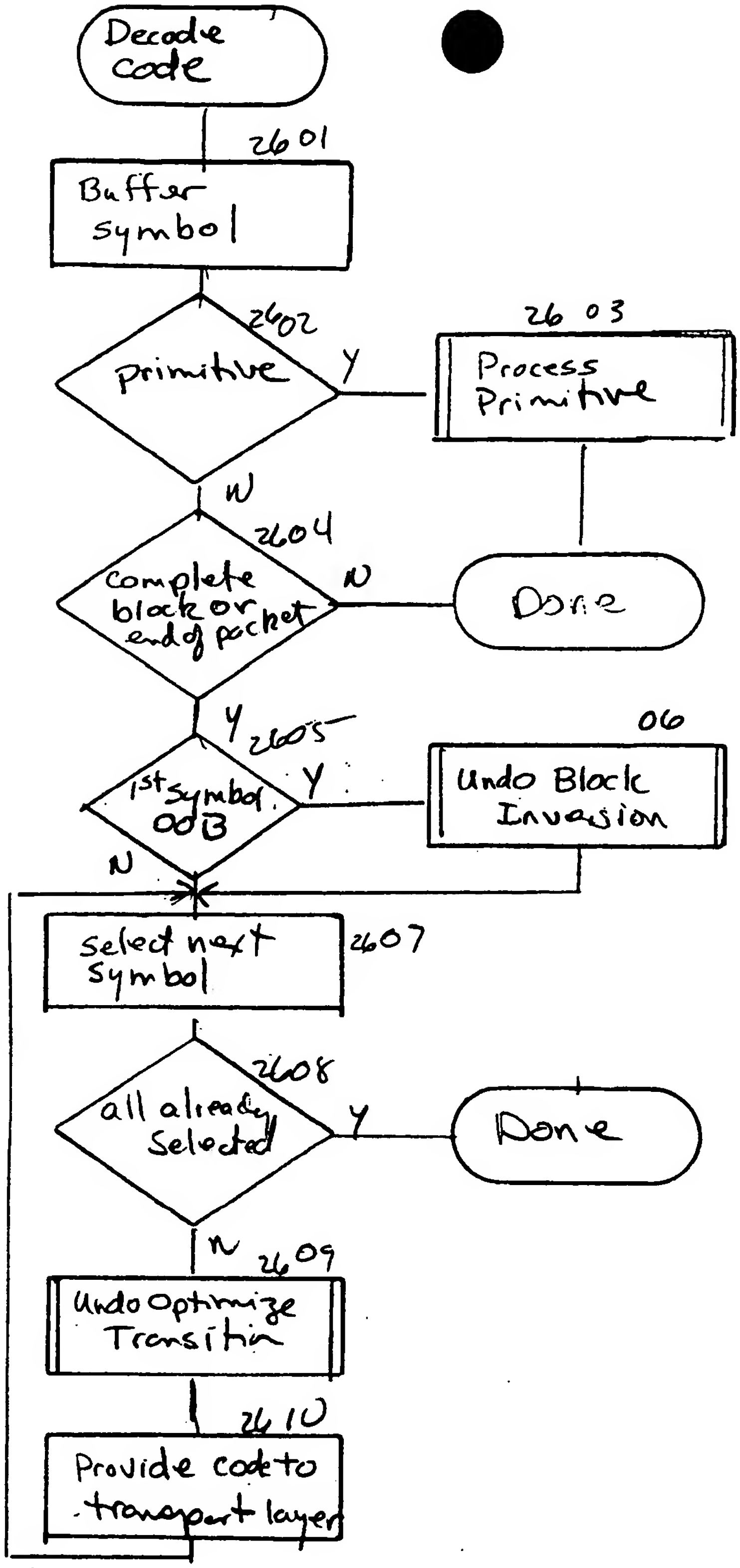

Decode code
2601
Buffer symbol
2602
primitive
Y
2603
Process Primitive
N
2604
complete block or end of packet
N
Done
Y
2605
06
1st symbol OOB
Y
Undo Block Invasion
N
select next symbol
2607
2608
all already selected
Y
Done
N
2609
Undo Optimize Transition
2610
Provide code to transport layer

Fig 26

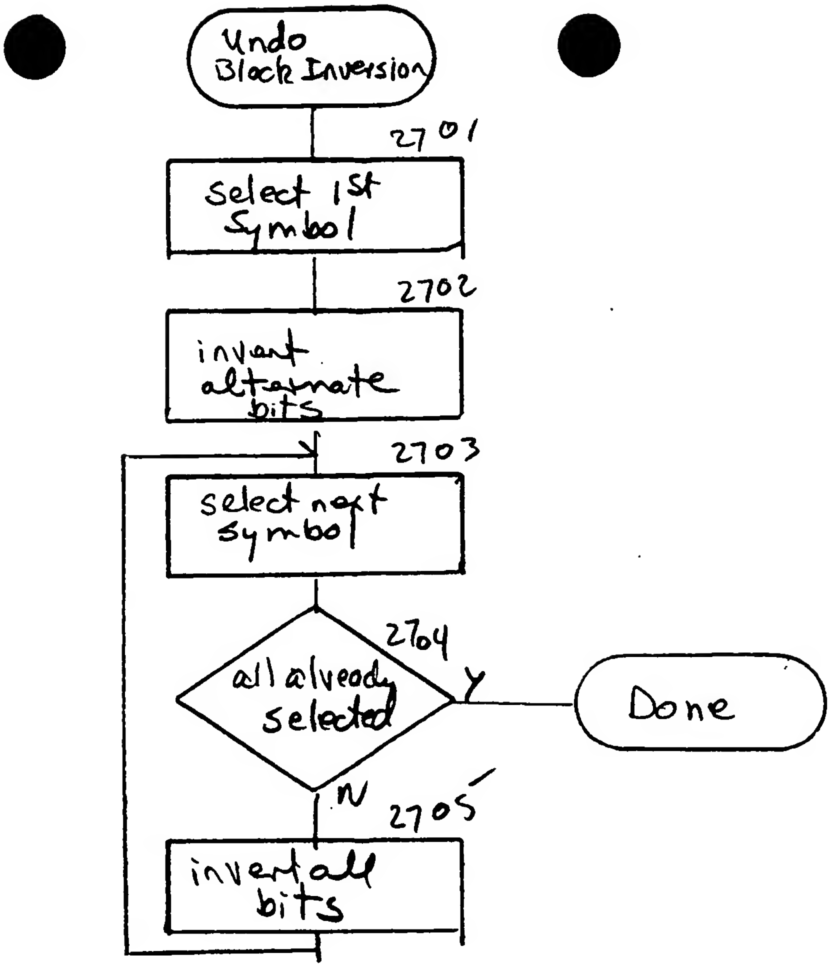

Undo Block Inversion
2701
select 1st Symbol
2702
invert alternate bits
2703
select next symbol
2704
all already selected
Y
Done
N
2705
invert all bits

Fig 27

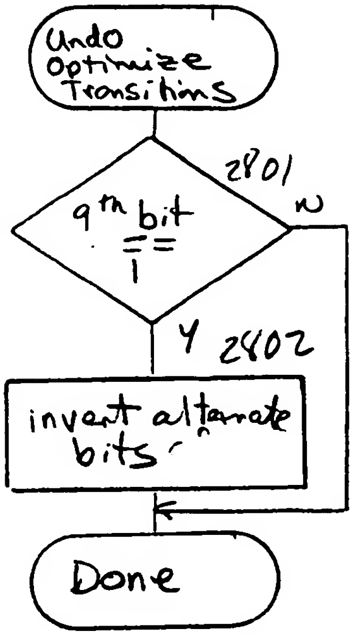

Undo Optimize Transitions
2801
9th bit == 1
N
Y
2802
invert alternate bits
Done

Fig 28

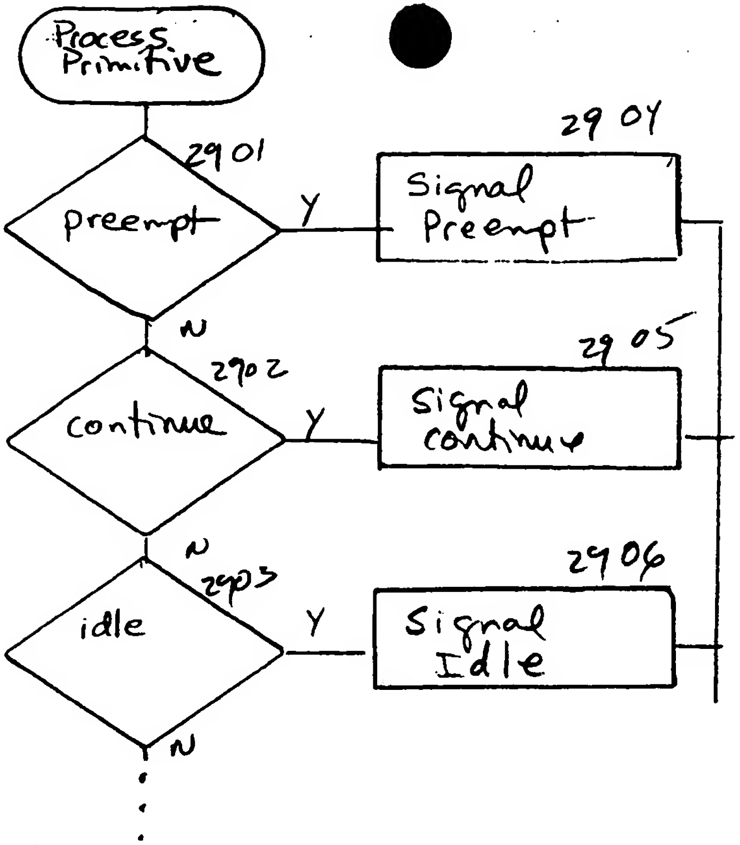

Process Primitive
2901
preempt
Y
N
2904
Signal Preempt
2902
continue
Y
N
2905
Signal continue
2903
idle
Y
N
2906
Signal Idle

Fig 29

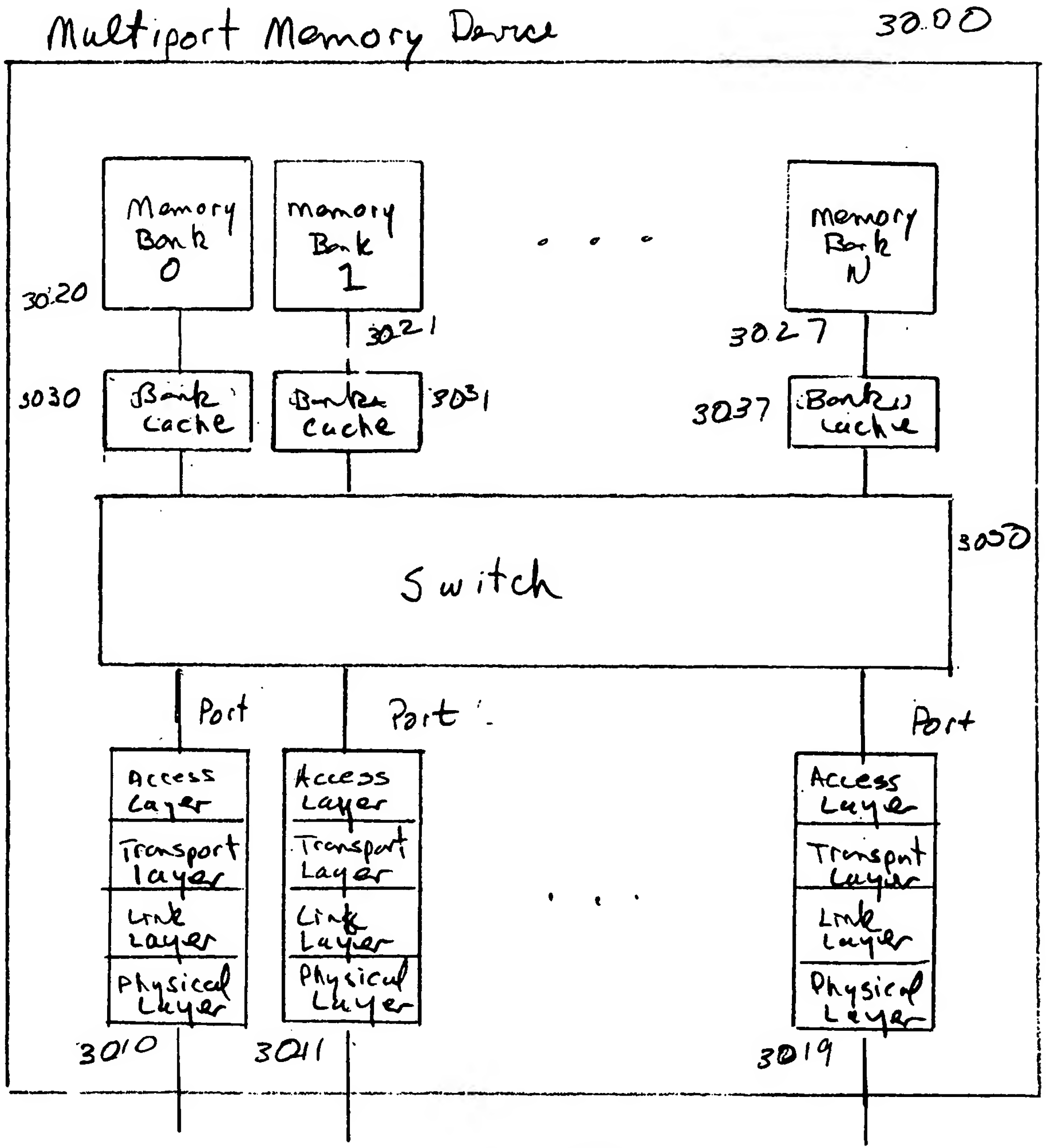

Multiport Memory Device
3000
Memory Bank 0
memory Bank 1
memory Bank N
3020
3021
3027
3030
Bank cache
Bank cache
3031
3037
Bank cache
Switch
3050
Port
Port
Port
Access Layer
Transport layer
Link Layer
Physical Layer
Access Layer
Transport Layer
Link Layer
Physical Layer
Access Layer
Transport Layer
Link Layer
Physical Layer
3010
3011
3019

Fig 30

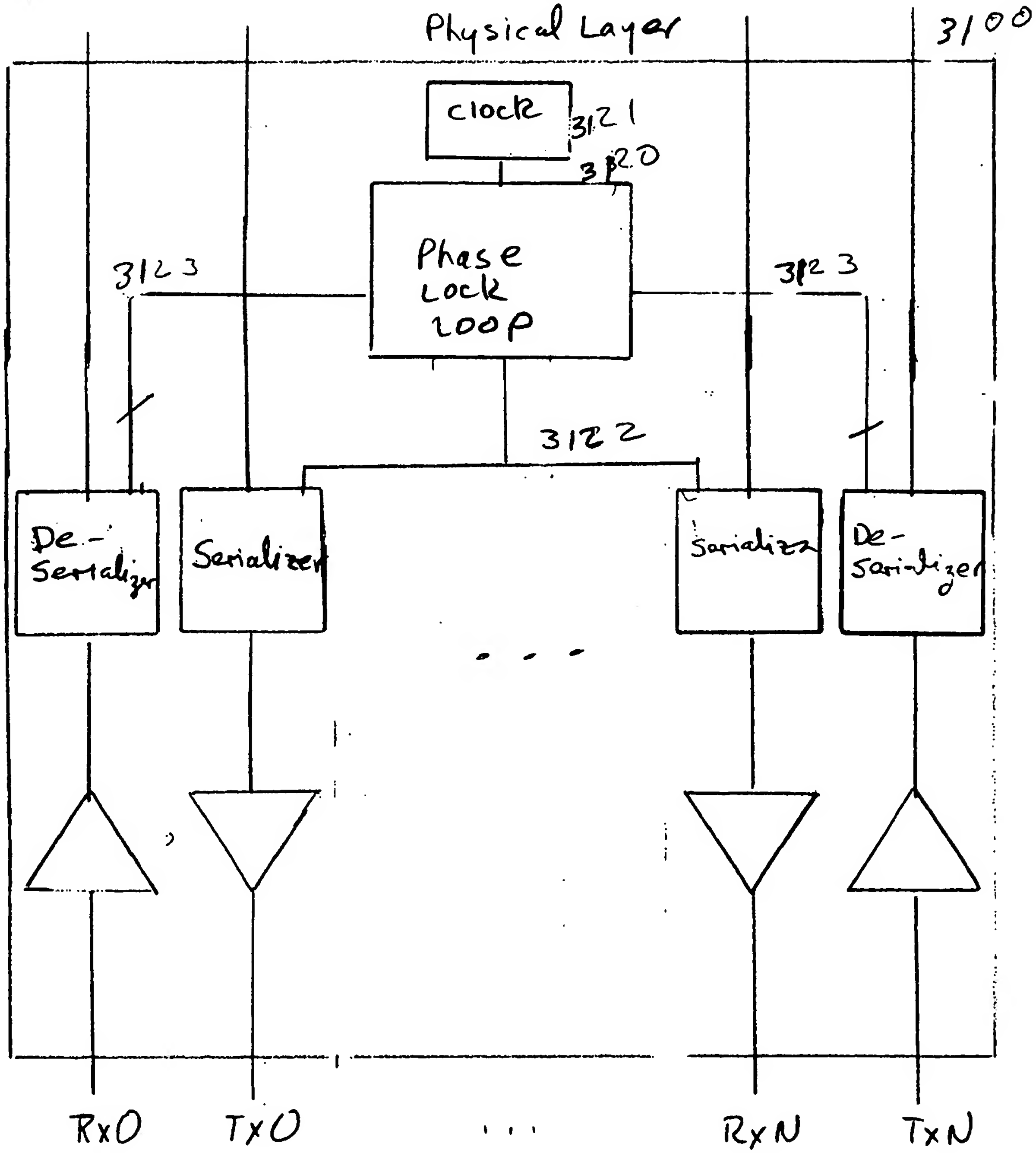

Physical Layer
3100
clock
3121
3120
Phase Lock Loop
3123
3123
3122
De-Serializer
Serializer
Serializer
De-Serializer
. . .
RxO
TxO
. . .
RxN
TxN
31.10
3119

Fig 31

Input Queue 3201

Port	R/W	Address	Data
3	R	1000	
4	W	4000	10....1
3	W	1000	111...0
3	R	2000	
	⋮		

Output Queue 3202

Valid	Port	Data
1	3	11...0
0		
0		
1	3	101...1
	⋮	

Fig 32

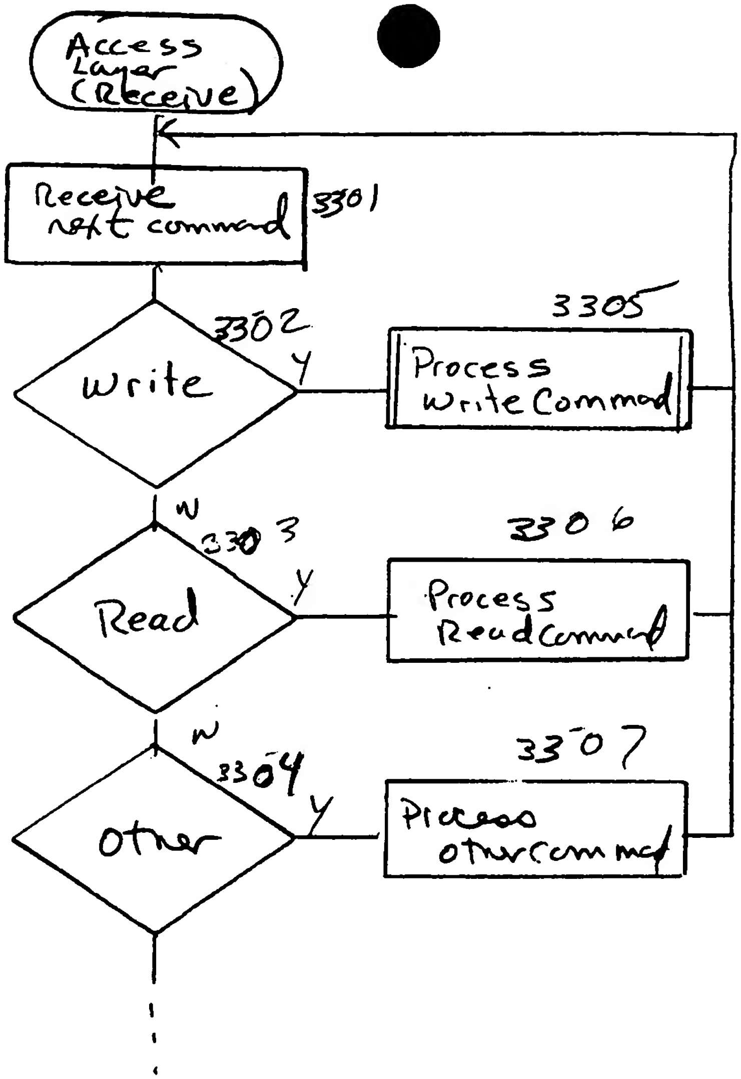

Access Layer (Receive)
Receive next command
3301
Write
3302
Y
Process Write Command
3305
N
Read
3303
Y
Process Read Command
3306
N
Other
3304
Y
Process other command
3307

Fig. 33

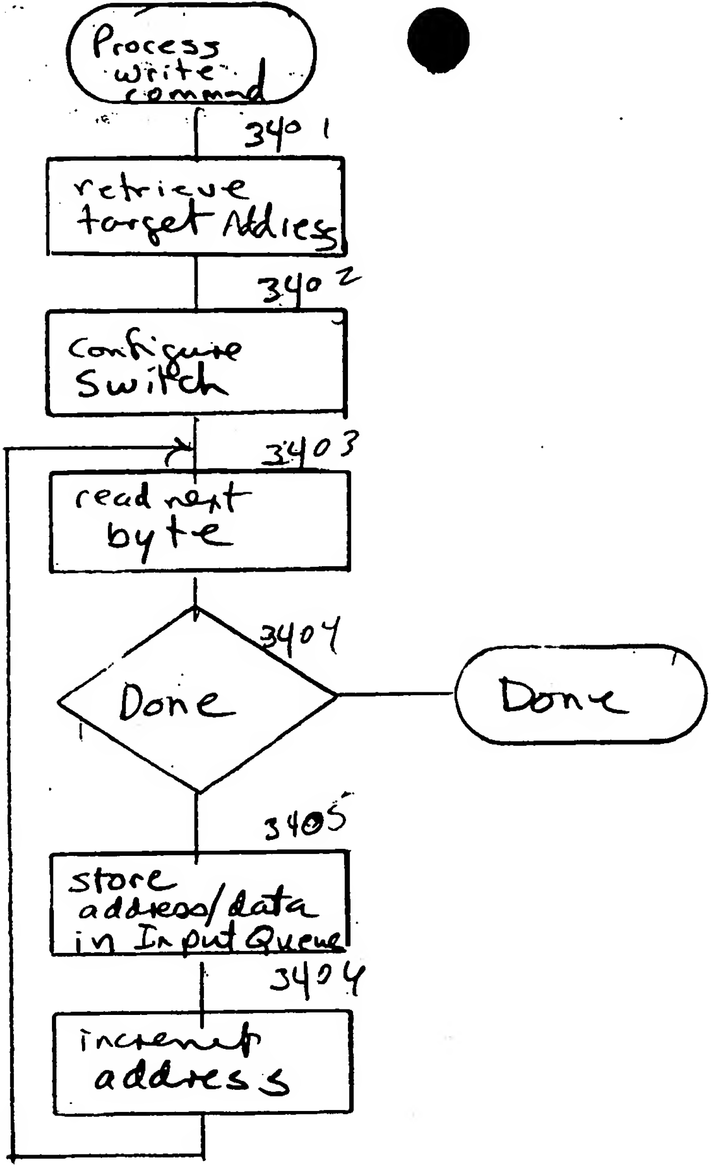

Process write command
3401
retrieve target Addiess
3402
Configure Switch
3403
read next byte
3404
Done
Done
3405
store address/data in Input Queue
3404
increment address

Fig 34

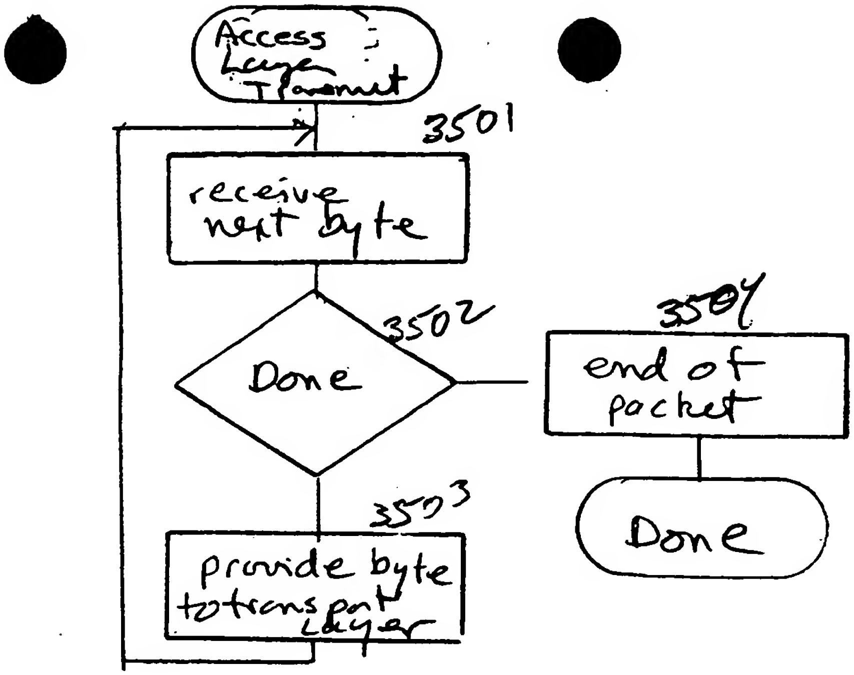

Access Layer Transmit
3501
receive next byte
3502
Done
3504
end of packet
Done
3503
provide byte to transport Layer

Fig 35

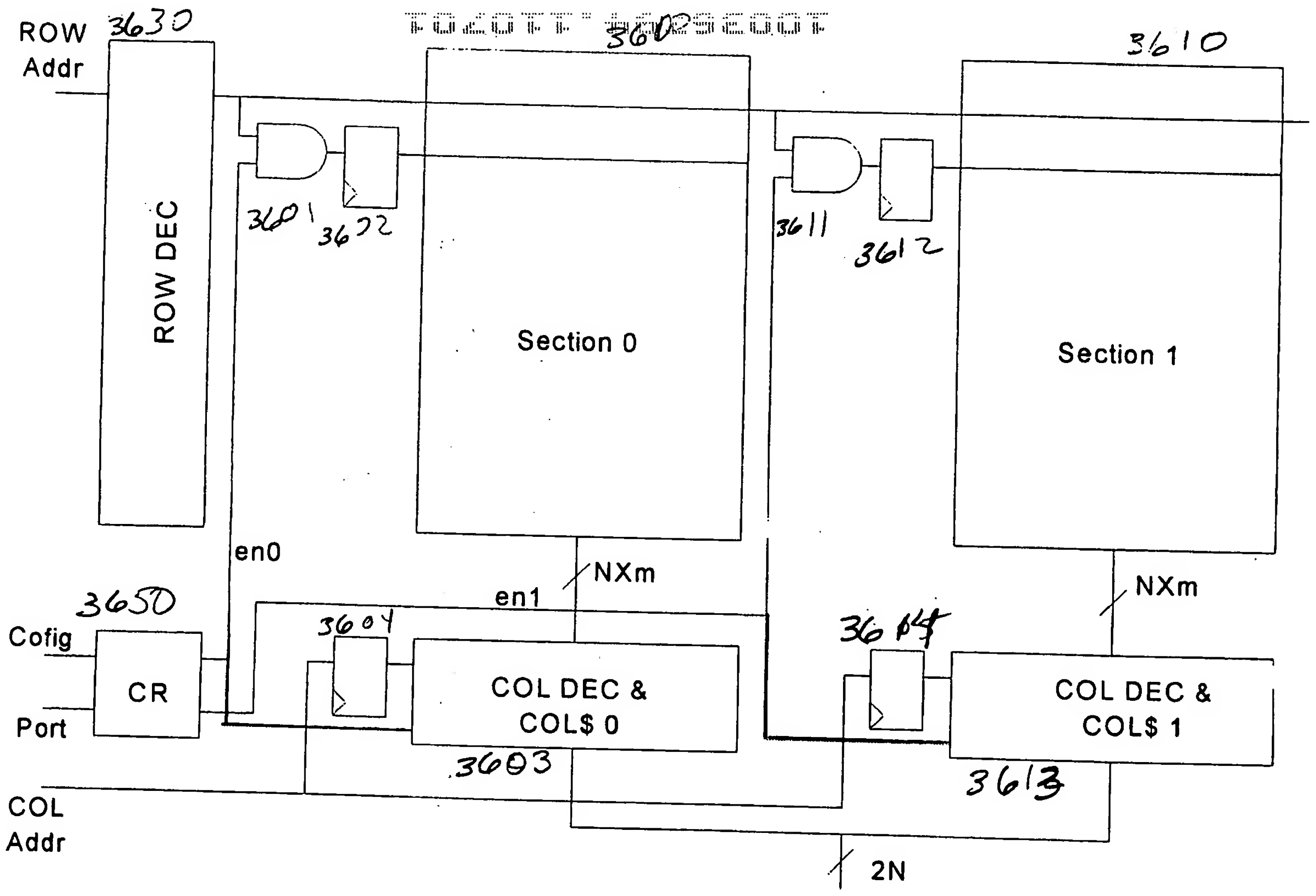
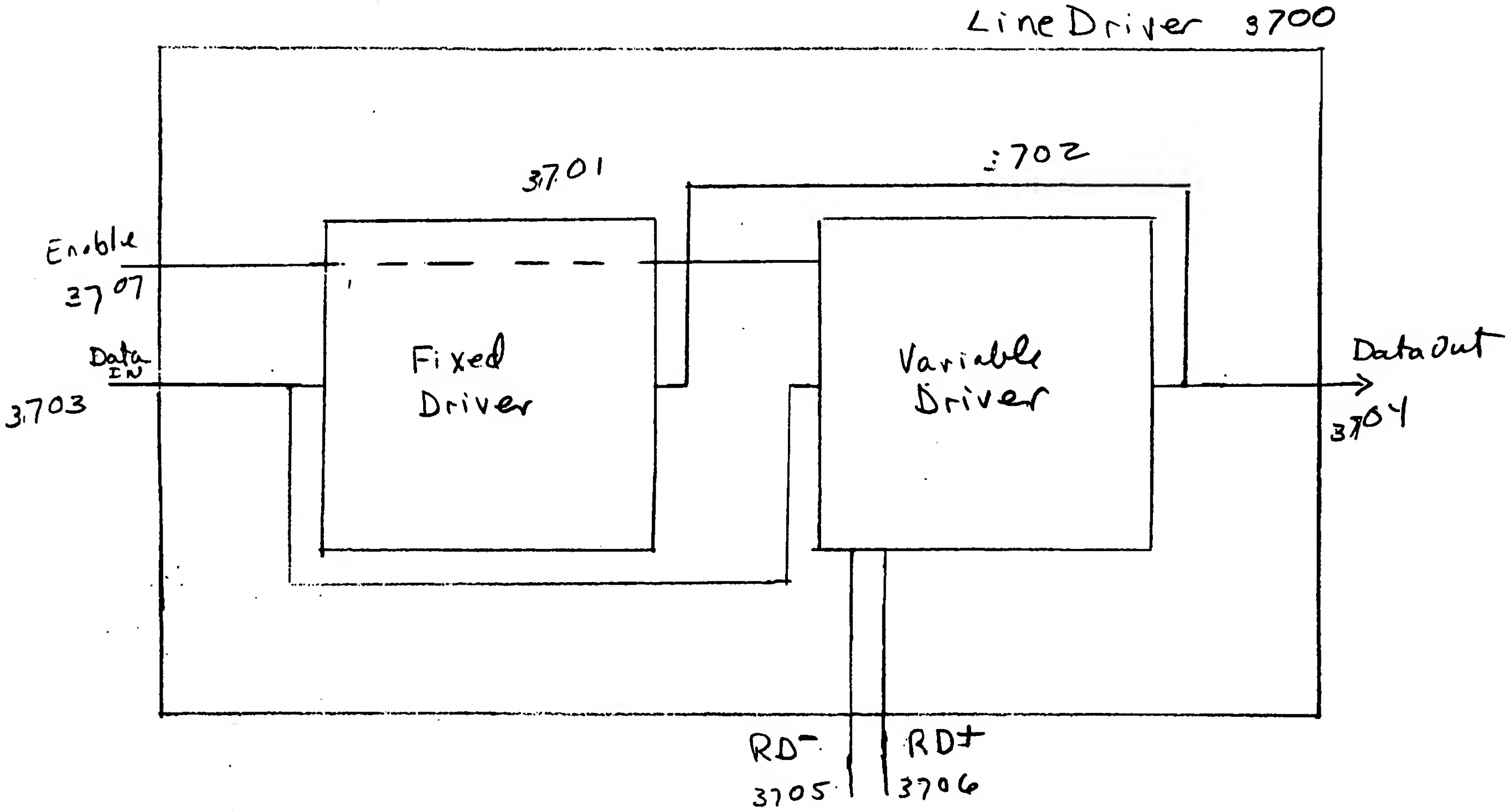

ROW
Addr
3630
ROW DEC
3601
3622
3600
Section 0
3611
3612
3610
Section 1
en0
en1
NXm
NXm
3650
Cofig
CR
Port
3604
COL DEC &
COL$ 0
3603
3614
COL DEC &
COL$ 1
3613
COL
Addr
2N

Fig 36



Variable Driver $\begin{cases} RD^{+} \wedge \overline{DataIn} = \text{pull down} \\ RD^{-} \wedge DataIn = \text{pull up} \end{cases}$

Fig 37A

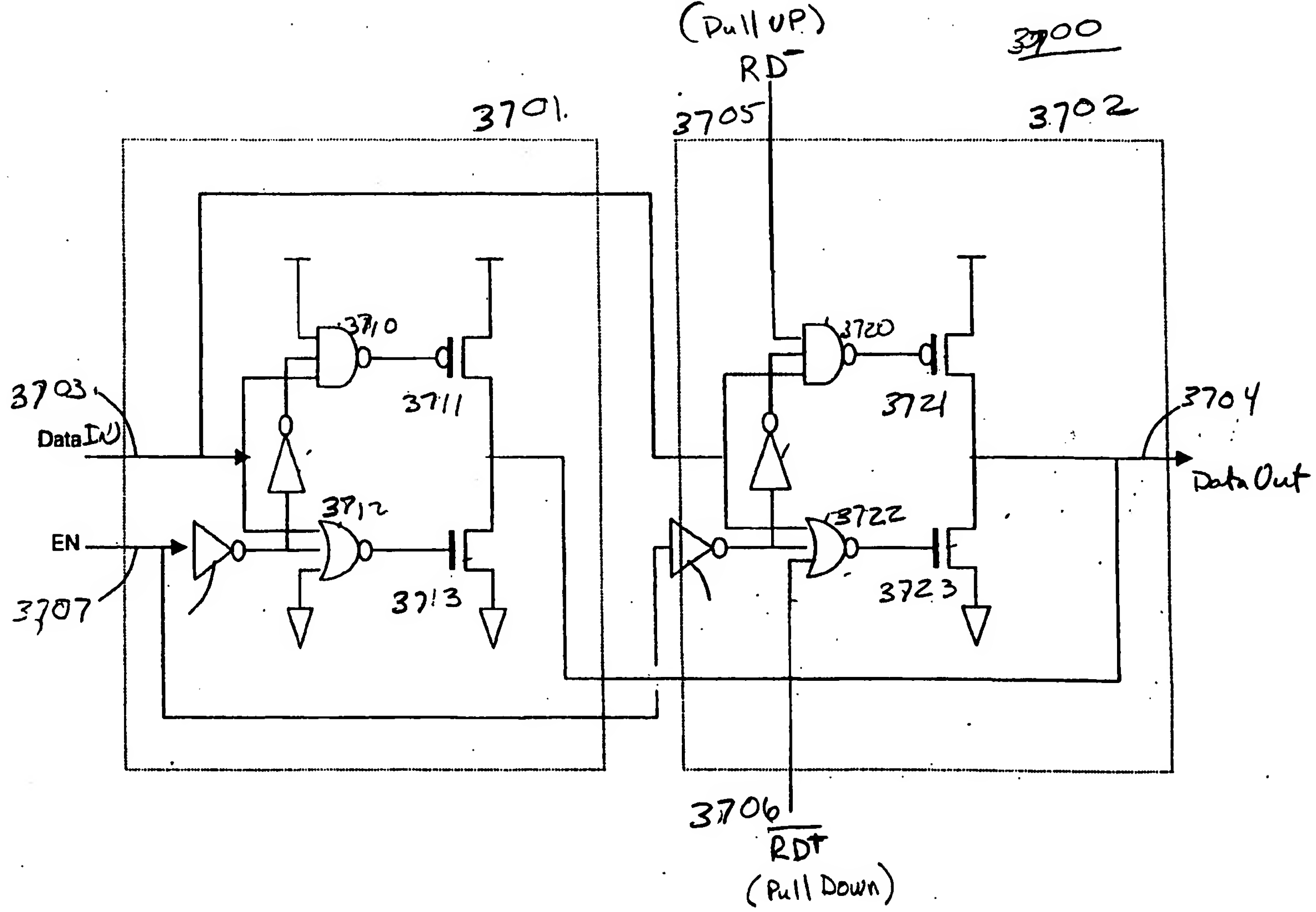

(Pull UP)
RD⁻
3700
3701.
3705
3702
3710
3720
3703
Data IN
3711
3721
3704
Data Out
3712
3722
EN
3707
3713
3723
3706
RD+
(Pull Down)

Fig 37B

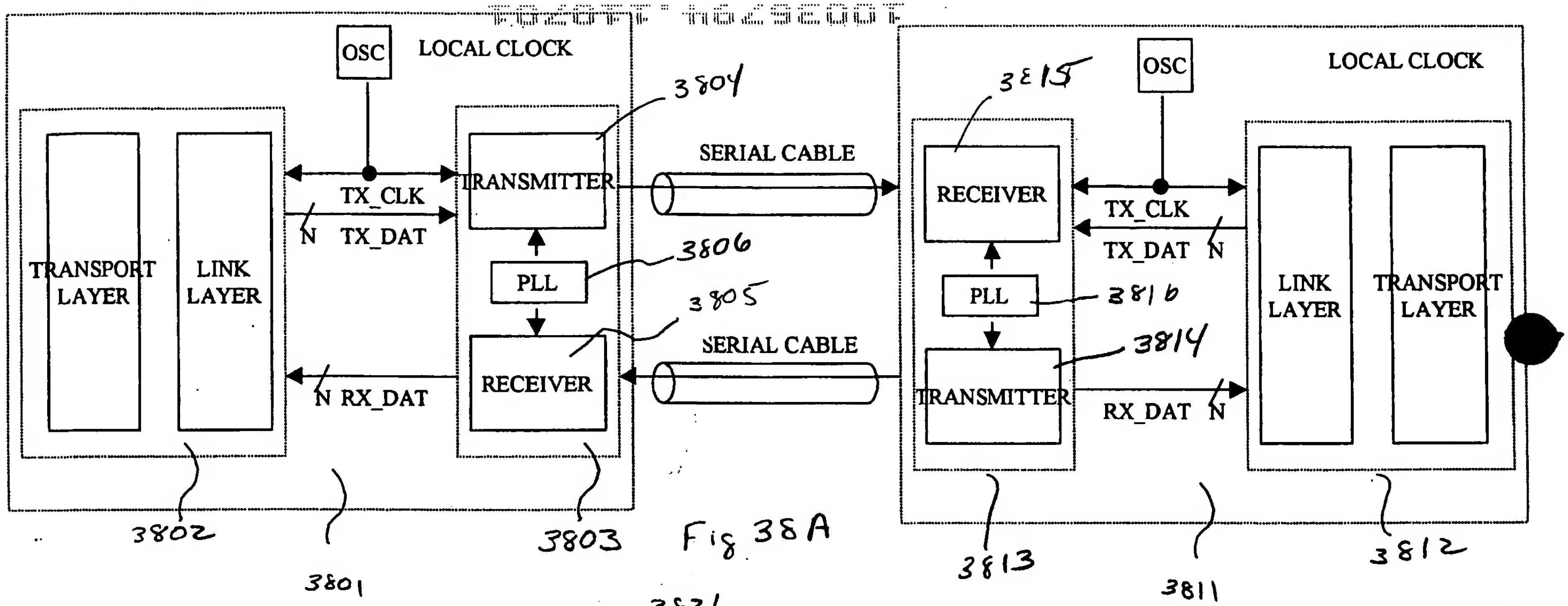


Fig 38A

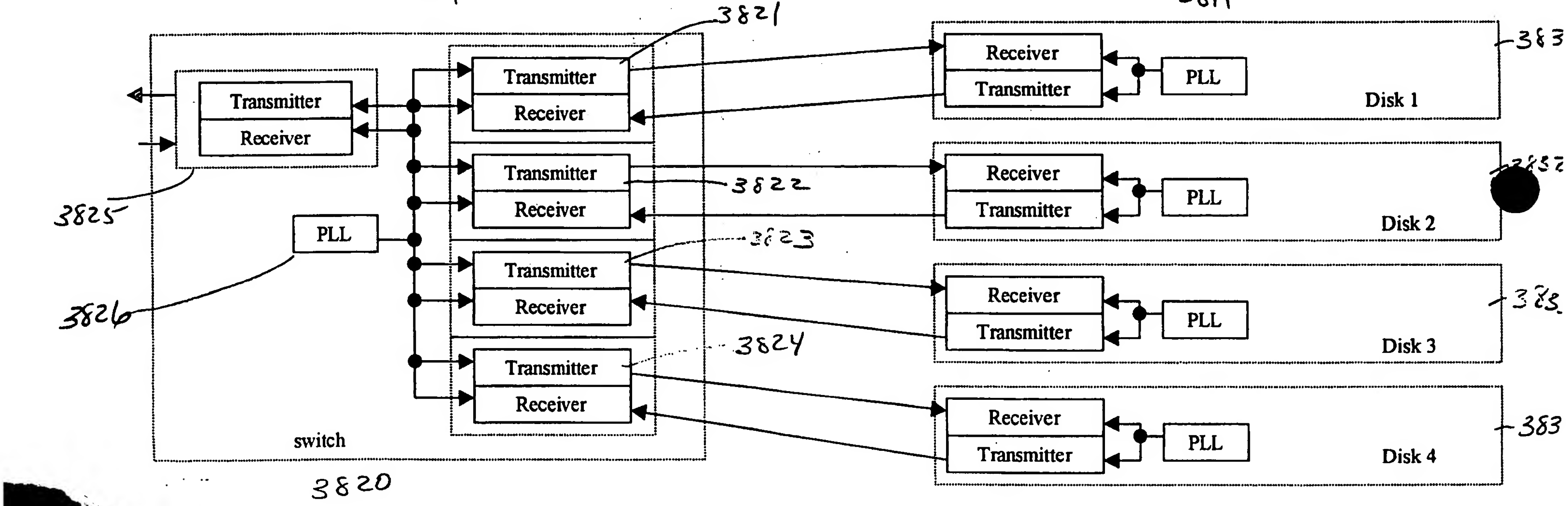


Fig 38B

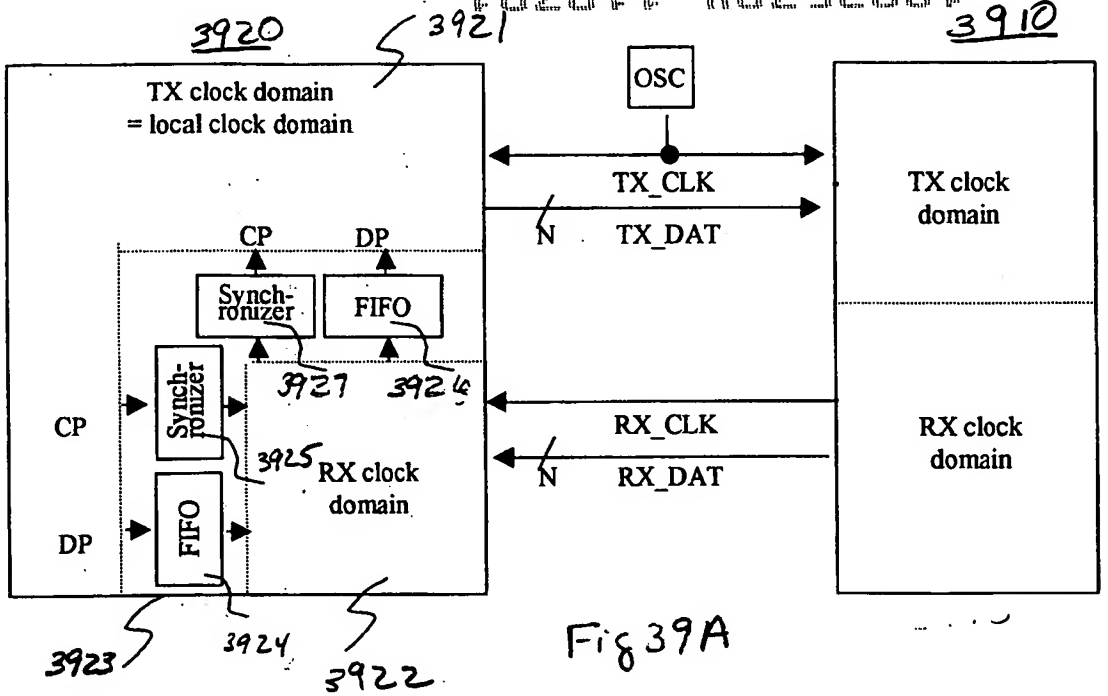


Fig 39A

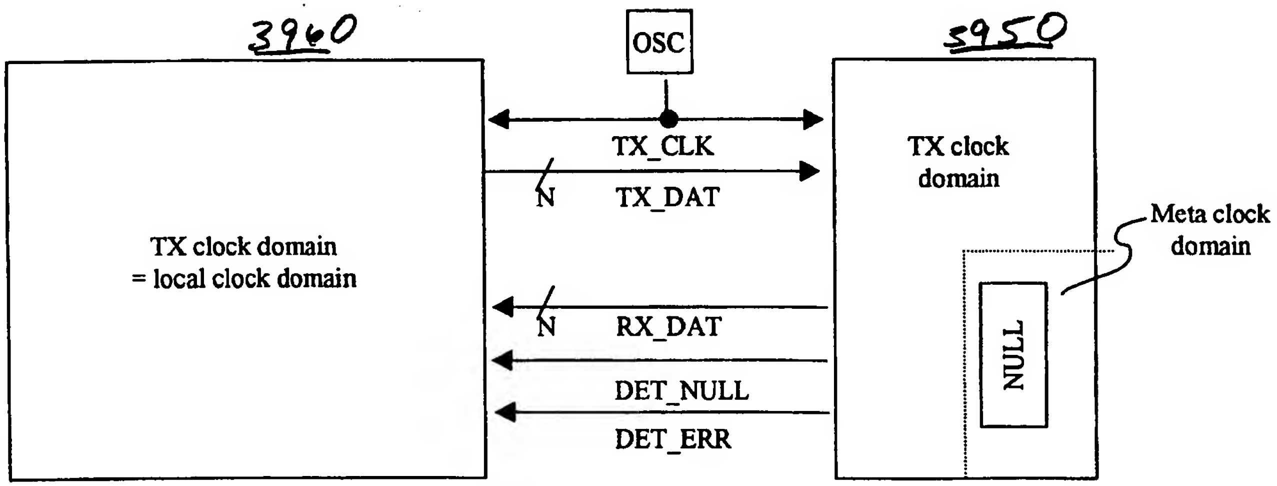


Fig 39B

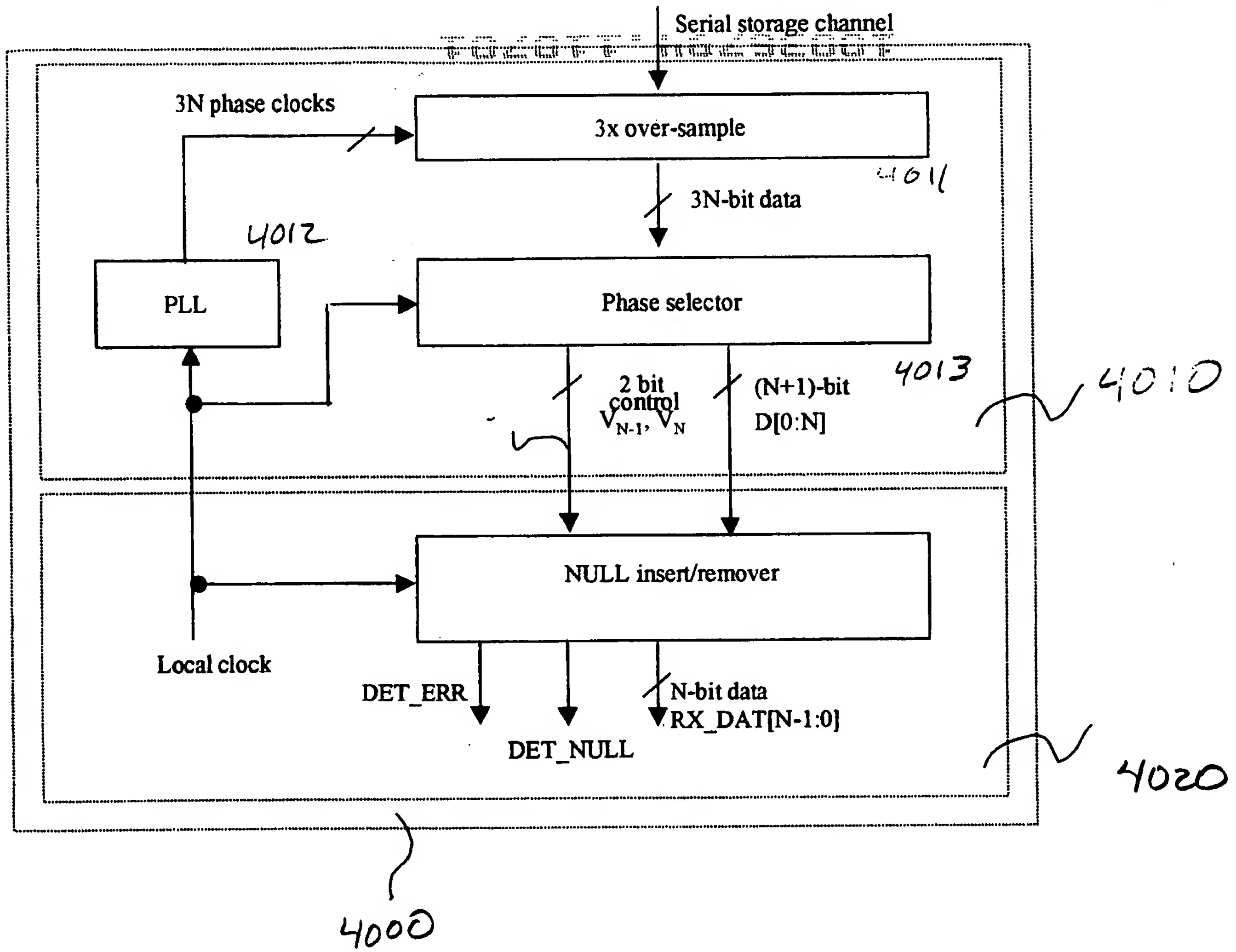

Serial storage channel
3N phase clocks
3x over-sample
4011
3N-bit data
4012
PLL
Phase selector
4013
4010
2 bit control
V_{N-1}, V_N
(N+1)-bit
D[0:N]
NULL insert/remover
Local clock
DET_ERR
DET_NULL
N-bit data
RX_DAT[N-1:0]
4020
4000

Fig 40

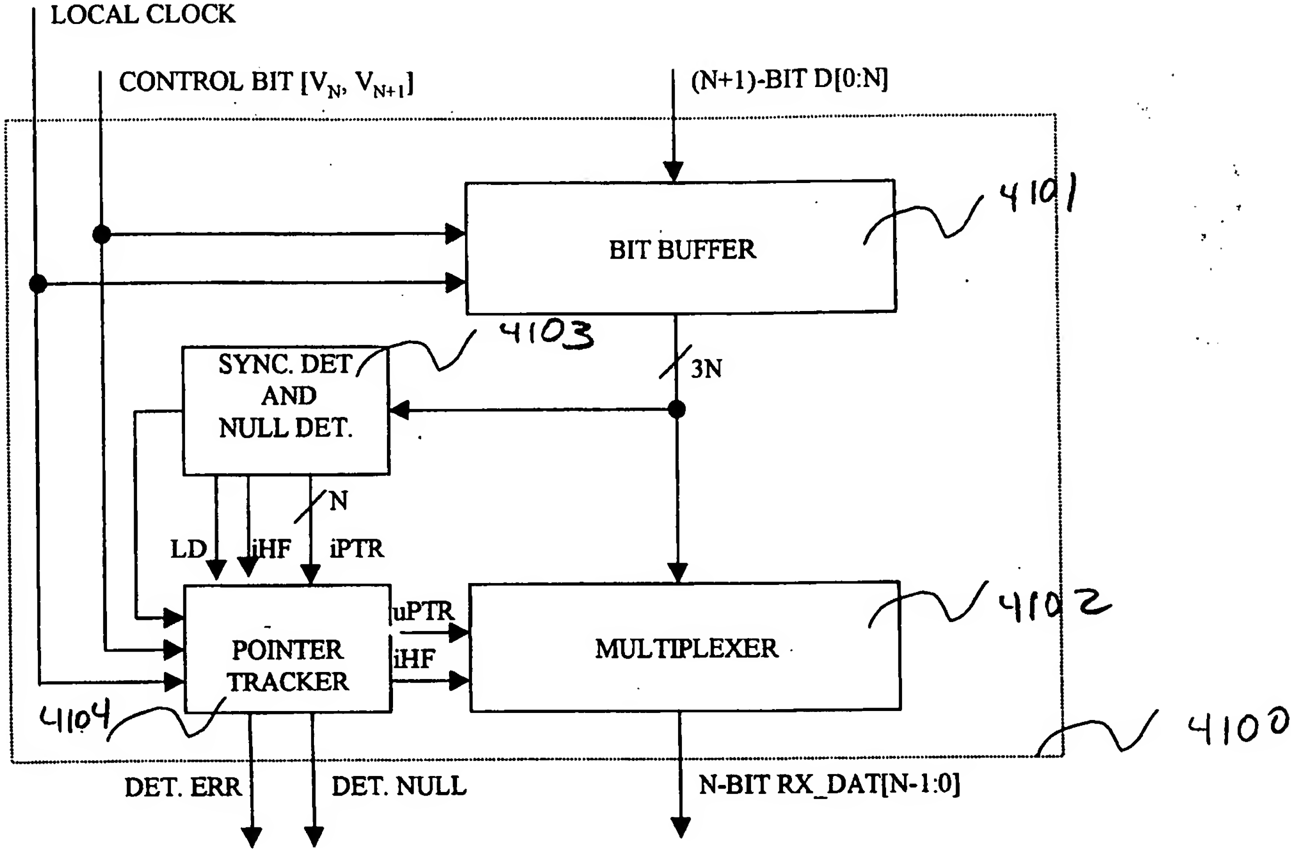

LOCAL CLOCK
CONTROL BIT [V_N, V_{N+1}]
(N+1)-BIT D[0:N]
BIT BUFFER
4101
4103
3N
SYNC. DET AND NULL DET.
N
LD
iHF
iPTR
POINTER TRACKER
uPTR
iHF
MULTIPLEXER
4102
4104
4100
DET. ERR
DET. NULL
N-BIT RX_DAT[N-1:0]

Fig 41

$[V_{N-1}, V_N] = [1,0]$

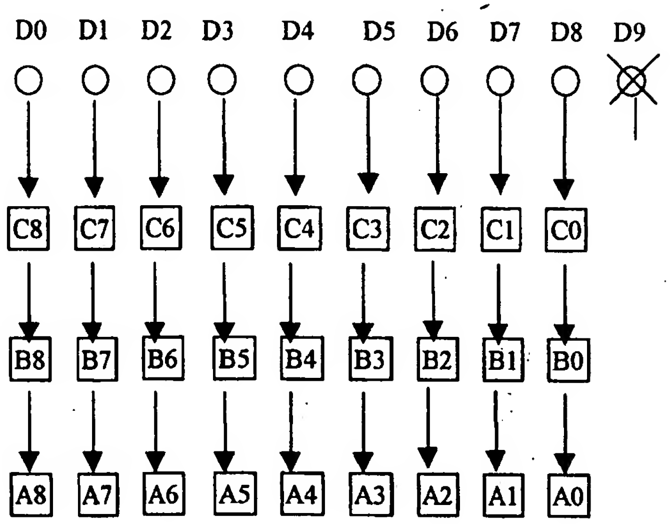


Fig 42A

$[V_{N-1}, V_N] = [0,0]$

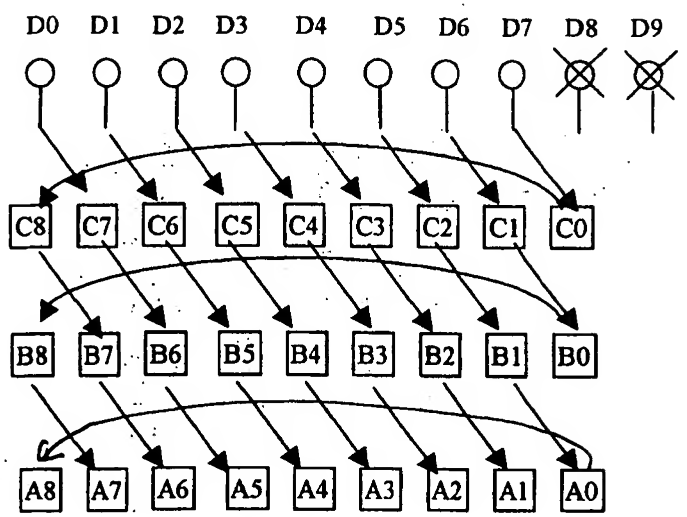


Fig 42B

$[V_{N-1}, V_N] = [1,1]$

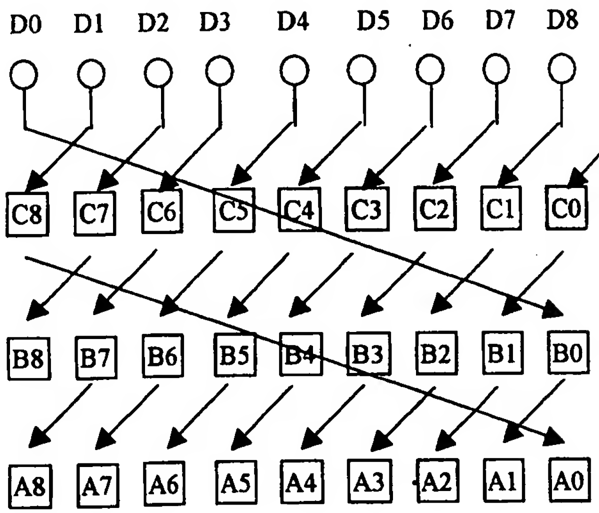


Fig 42C

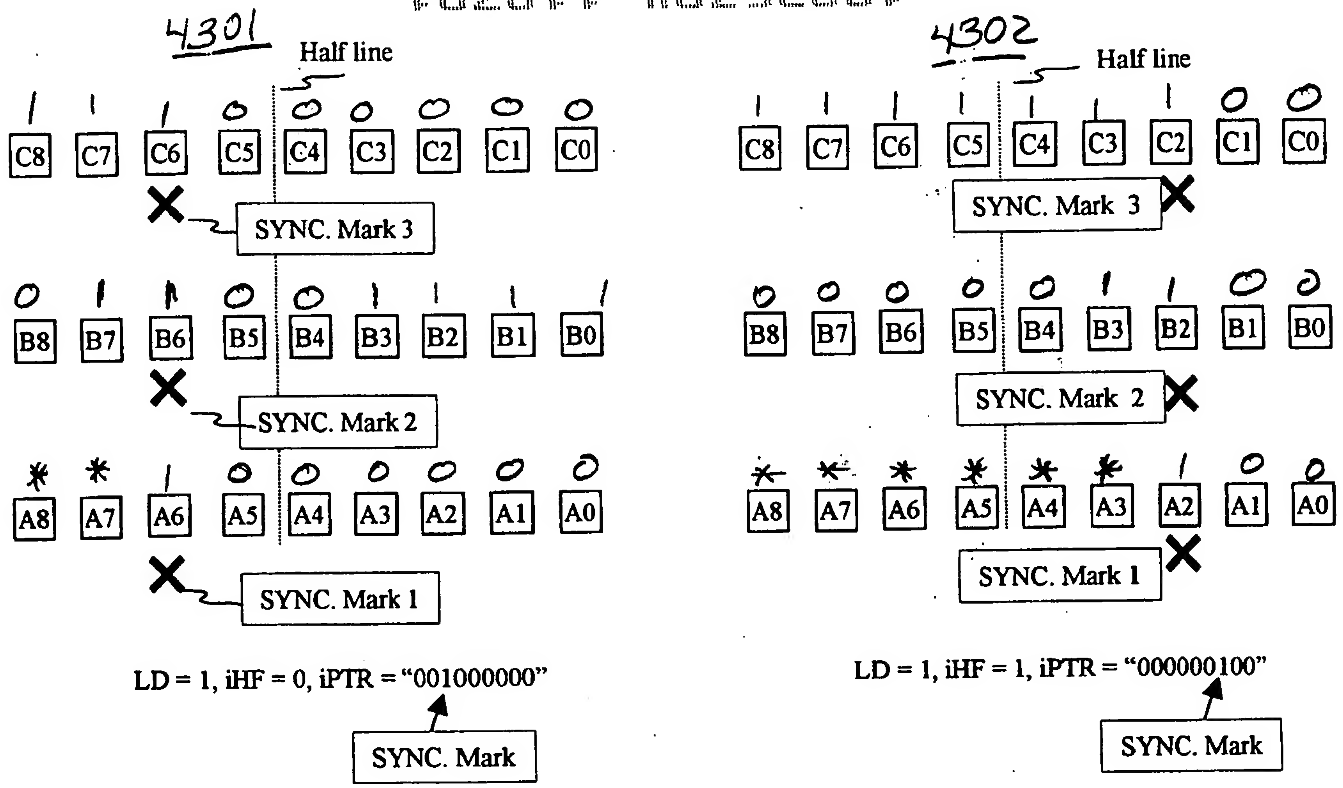
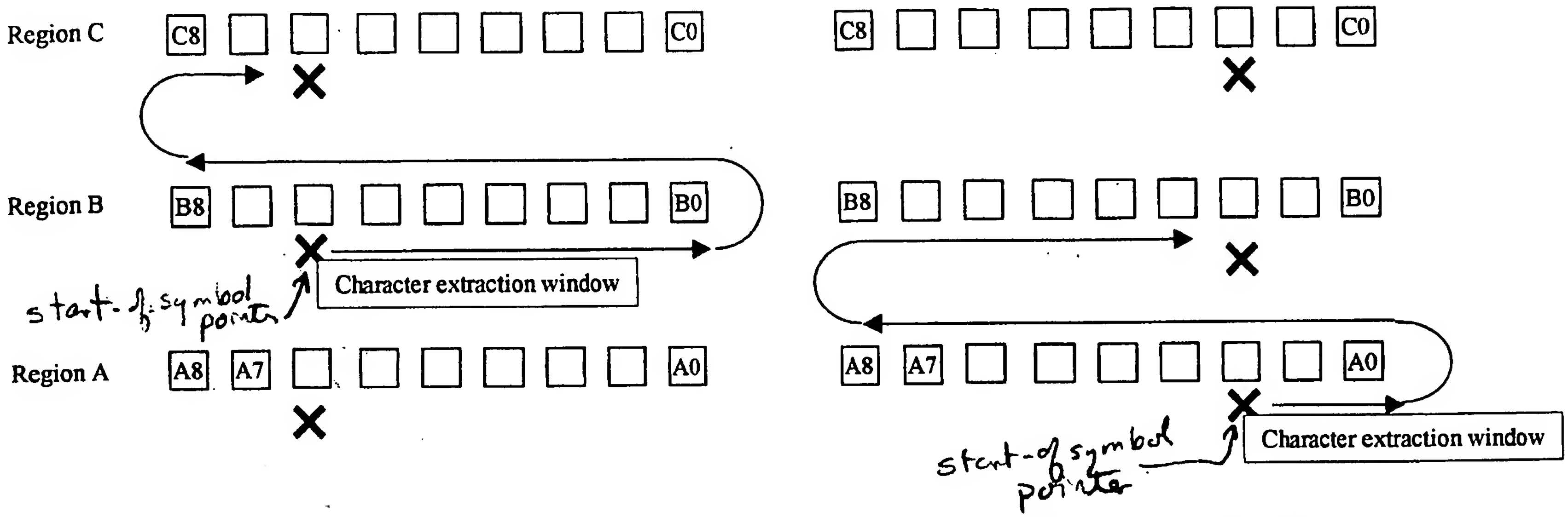

4301
Half line
1 1 1 0 0 0 0 0 0
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
0 1 1 0 0 1 1 1 1
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
* * 1 0 0 0 0 0 0
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 0, iPTR = "001000000"
SYNC. Mark
4302
Half line
1 1 1 1 1 1 1 0 0
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
0 0 0 0 0 1 1 0 0
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
* * * * * * 1 0 0
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 1, iPTR = "000000100"
SYNC. Mark

Fig. 43



LD = 1, iHF = 0, iPTR = “001000000”

LD = 1, iHF = 1, iPTR = “000000100”

Fig 44

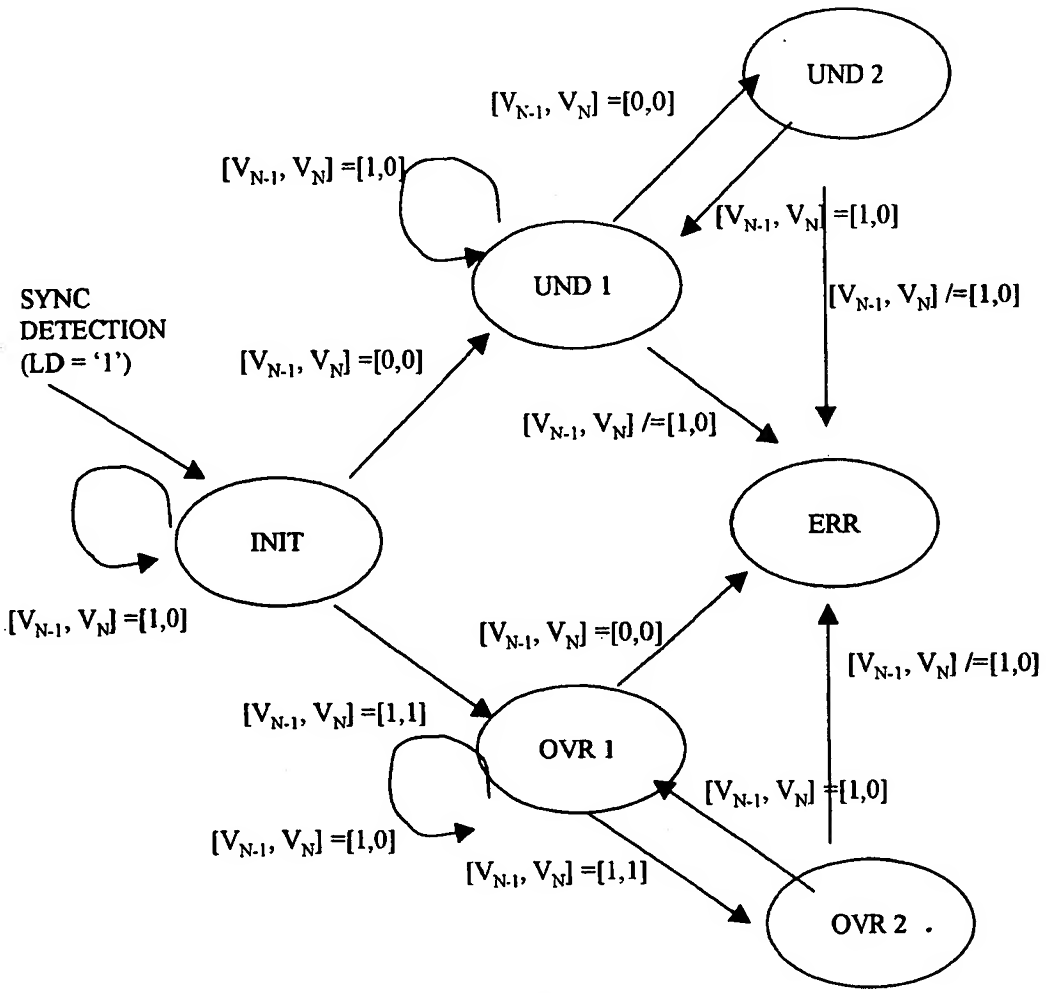

UND 2
UND 1
INIT
ERR
OVR 1
OVR 2 .
SYNC DETECTION (LD = ‘1’)
[V$_{N-1}$, V$_N$] =[0,0]
[V$_{N-1}$, V$_N$] =[1,0]
[V$_{N-1}$, V$_N$] =[1,0]
[V$_{N-1}$, V$_N$] /=[1,0]
[V$_{N-1}$, V$_N$] =[0,0]
[V$_{N-1}$, V$_N$] /=[1,0]
[V$_{N-1}$, V$_N$] =[1,0]
[V$_{N-1}$, V$_N$] =[0,0]
[V$_{N-1}$, V$_N$] /=[1,0]
[V$_{N-1}$, V$_N$] =[1,1]
[V$_{N-1}$, V$_N$] =[1,0]
[V$_{N-1}$, V$_N$] =[1,0]
[V$_{N-1}$, V$_N$] =[1,1]

Fig 45

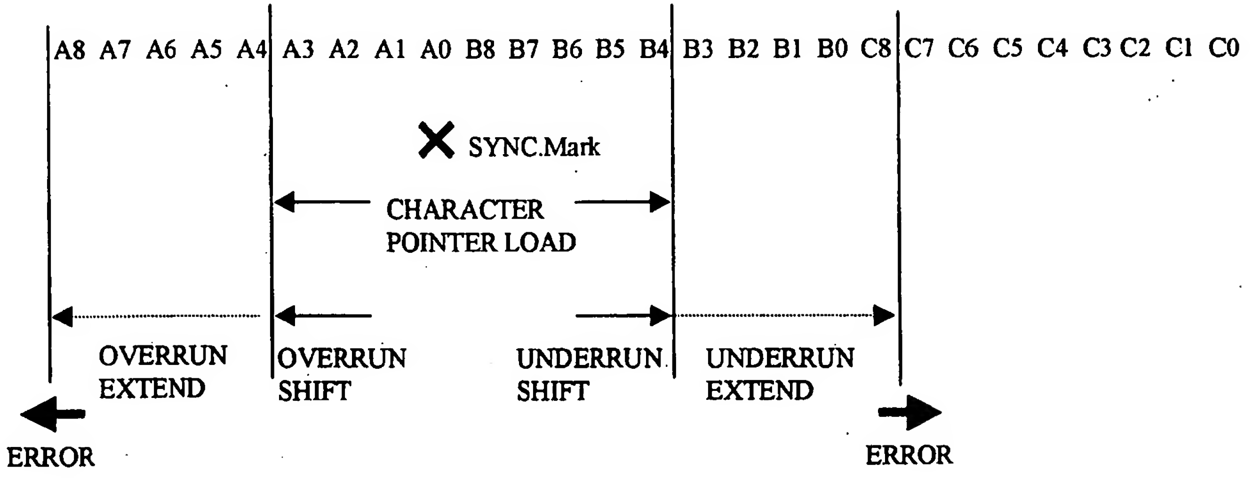

A8 A7 A6 A5 A4 A3 A2 A1 A0 B8 B7 B6 B5 B4 B3 B2 B1 B0 C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC.Mark
CHARACTER POINTER LOAD
OVERRUN EXTEND
OVERRUN SHIFT
UNDERRUN SHIFT
UNDERRUN EXTEND
ERROR
ERROR

Fig 46

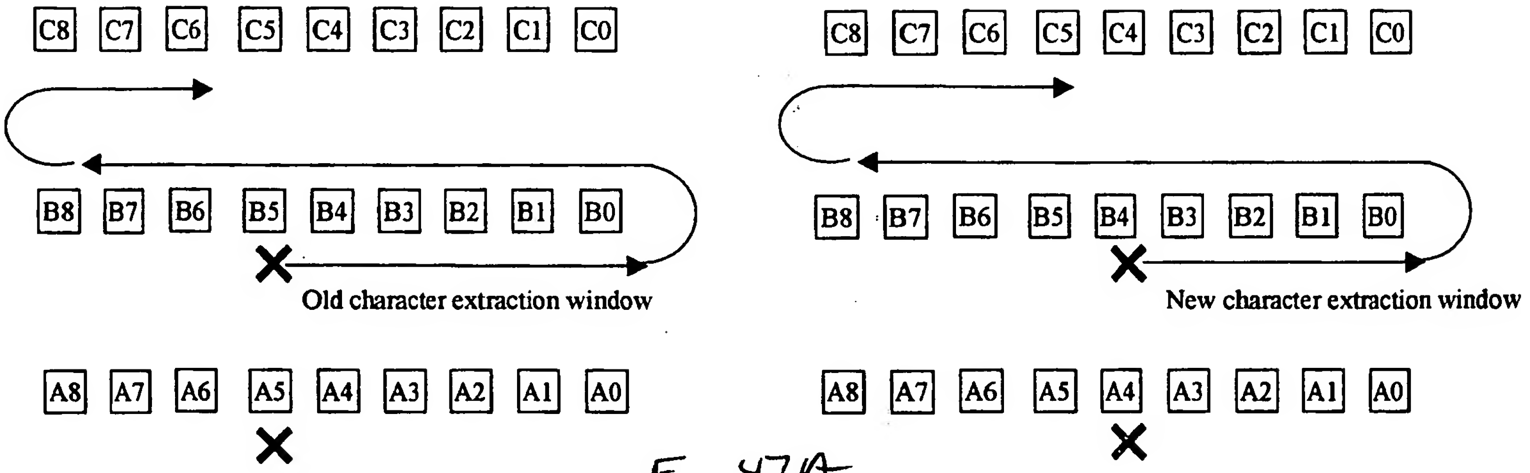


Fig 47A

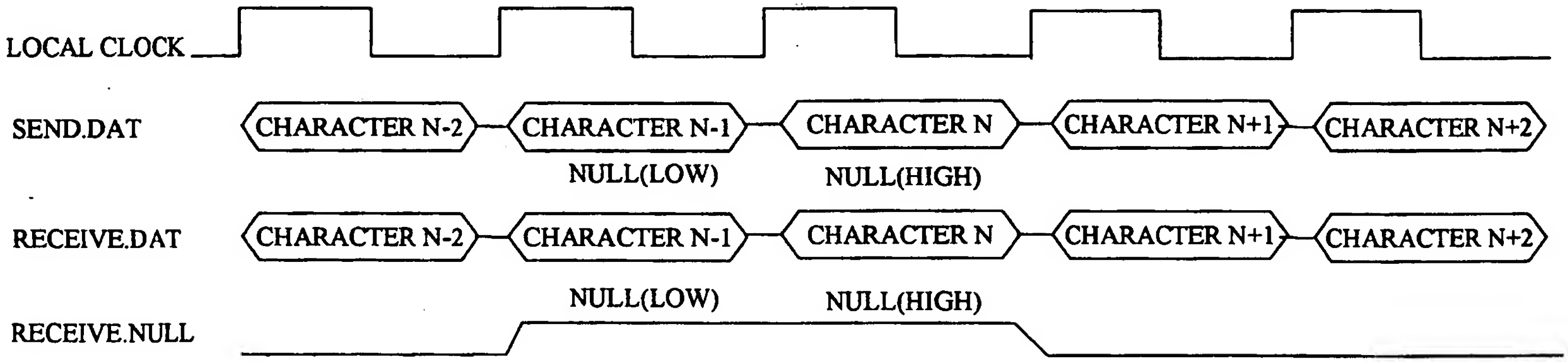


Fig 47B

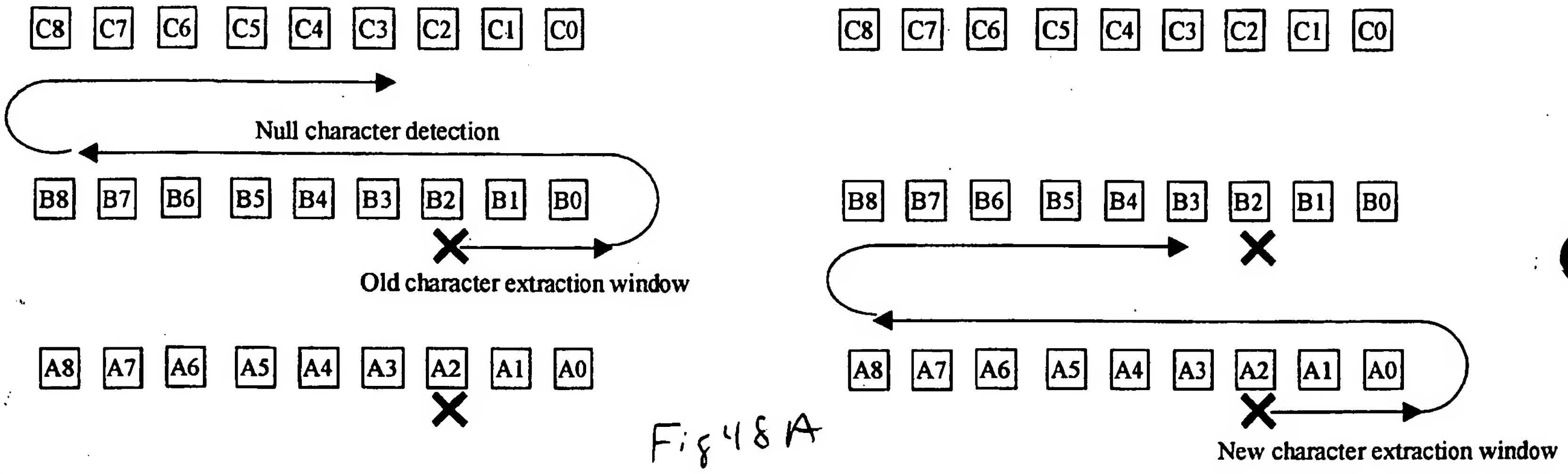


Fig 48A

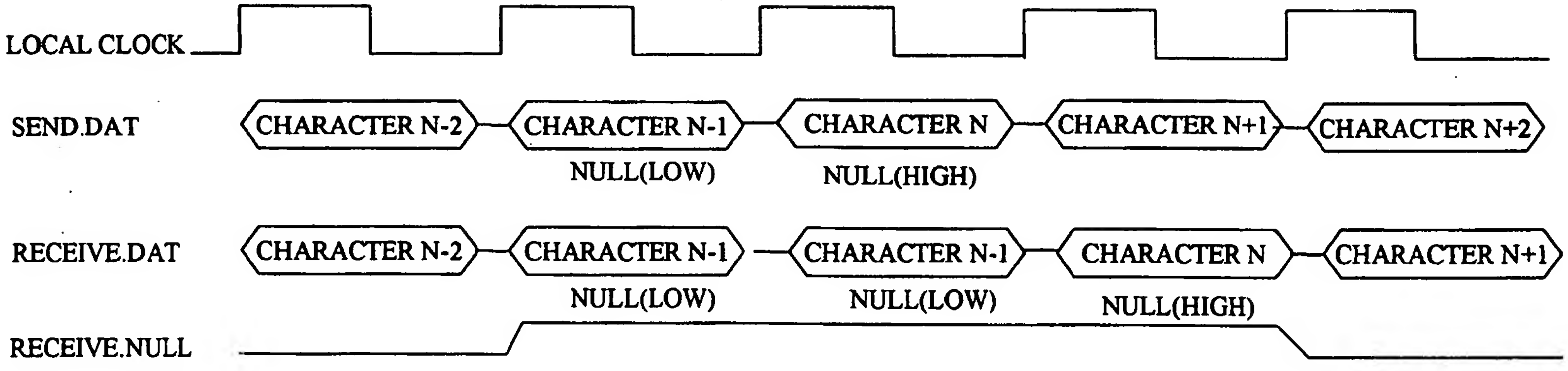


Fig 48B

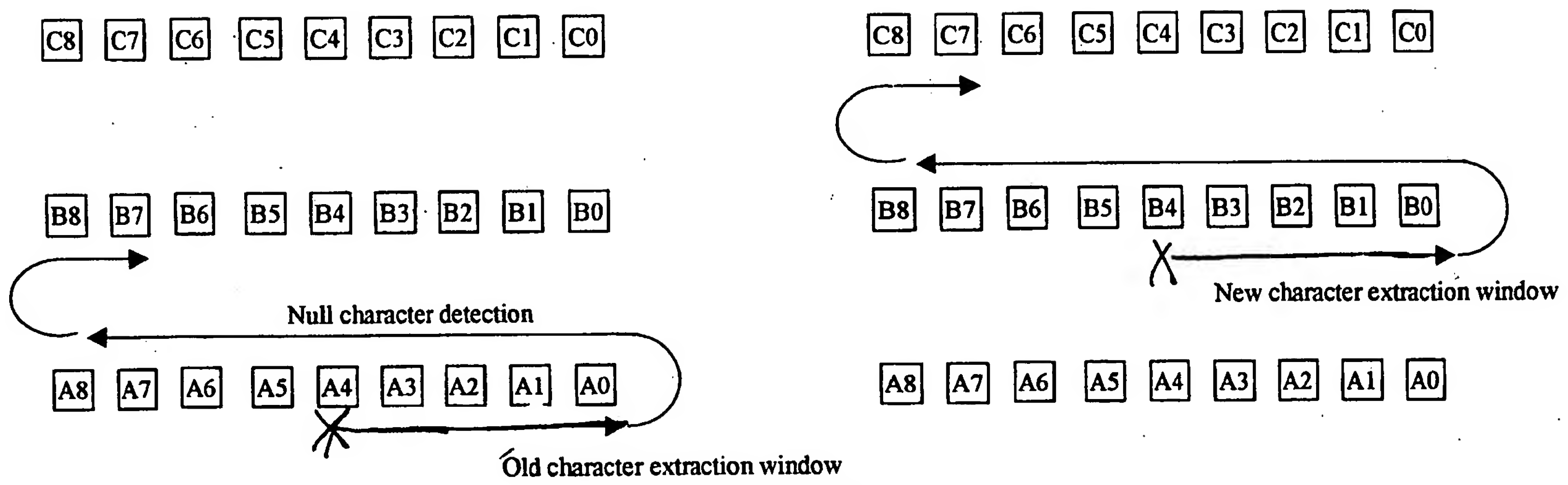


Fig. 49A

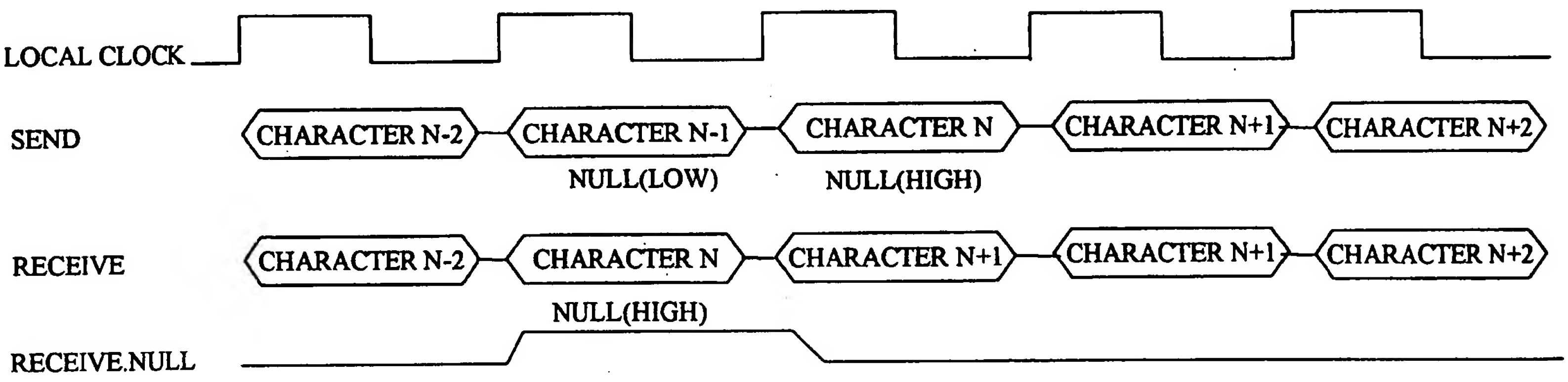


Fig 49B